CSA GCR cloud security alliance®
GREATER CHINA REGION

共识算法与共识安全白皮书

致 谢

云安全联盟大中华区（简称：CSA GCR）区块链安全工作组在 2020 年 2 月份成立。由黄连金担任工作组组长，9 位领军人分别担任 9 个项目小组组长，分别有：知道创宇创始人&CEO 赵伟领衔数字钱包安全小组，北大信息科学技术学院区块链研究中心主任陈钟领衔共识算法安全小组，赛博英杰创始人&董事长谭晓生领衔交易所安全小组，安比实验室创始人郭宇领衔智能合约安全小组，世界银行首席信息安全架构师张志军领衔 Dapp 安全小组，中国移动集团信安中心高工于乐博士领衔去中心化数字身份安全小组，北理工教授祝烈煌领衔网络层安全小组，武汉大学教授陈晶领衔数据层安全小组，零时科技 CEO 邓永凯领衔 AML 技术与安全小组。区块链安全工作组现有 100 多位安全专家们，分别来自中国电子学会、耶鲁大学、北京大学、北京理工大学、世界银行、中国金融认证中心、华为、腾讯、知道创宇、慢雾科技、启明星辰、天融信、联想、OPPO、零时科技、普华永道、安永、阿斯利康等六十多家单位。

本白皮书主要由共识算法安全小组专家撰写，感谢以下专家的贡献：

原创作者：陈 钟　关 志　王 珂

审核专家：黄连金　江 洪　郭鹏程　姚 凯

关于研究工作组的更多介绍，请在 CSA 大中华区官网（https://c-csa.cn/research/）上查看。

如本白皮书有不妥当之处，敬请读者联系 CSA GCR 秘书处给与雅正! 联系邮箱：research@c-csa.cn; 国际云安全联盟 CSA 公众号。

序 言

从早期的分布式一致性算法的缓慢发展到现如今区块链共识的百花齐放，共识算法的发展已经走过了四十年左右的时光。随着区块链技术的快速发展，共识算法也在不断演进和提高，不同的共识算法的侧重点不同，因此它们所面临的问题、环境也不一样。

共识算法，可以理解为是为了实现分布式一致性协议而产生的一系列流程与规则。当分布在不同地域的节点都按照这套规则进行协商交互之后，最终总能就某个问题得到一致的决策，从而实现分布式系统中不同节点的一致性。

《共识算法与共识安全》白皮书深入介绍了 CFT 类共识算法、经典拜占庭共识和开放 BFT 类共识等三大类共识算法，并延伸到了共识算法安全性分析和测试方法。报告旨在提供一份系统的关于共识算法安全的知识，从共识算法理论和实现两方面展开安全测试和分析，理论分析主要从算法本身展开分析，实现分析从算法的具体参数、代码实现、应用部署等都方面进行安全分析和测试。基于测试方法和标准，对共识算法安全的典型安全进行分析，并给出分析过程和结论。报告以超级账本和以太坊共识算法为例，探索这些项目在共识算法安全领域的实践经验。

报告旨在为区块链开发者、投资者、组织机构在共识算法安全领域提供指导，通过详细数据和案例进行论证，力求将理论与实践更好结合，希望能为您提供有关共识算法的有益信息，帮助您更好地理解和应用这项技术。

李雨航 Yale Li

CSA 大中华区主席兼研究院院长

目录

1 共识算法介绍

1.1 介绍

共识问题是社会科学和计算机科学等领域的经典问题，计算机科学领域的早期共识研究一般聚焦于分布式一致性，即如何保证集群中所有节点中的数据完全相同并且能够对某个提案（Proposal）达成一致，主要解决的问题是因节点失效、节点间网络故障及分布式系统的运行速度的差异而带来的系统不一致问题。Fischer 等在 1985 年提出了 FLP 不可能原理，即在网络可靠，但允许节点失效（即便只有一个）的最小化异步模型系统中，不存在一个可以解决一致性问题的确定性共识算法[1]。Eric Brewer 在 1998 年提出了 CAP 理论，指出在异步的网络模型中，所有的节点由于没有时钟，仅仅能根据接收到的消息作出判断，系统最多只能同时满足一致性（Consistency）、可用性（Availability）和分区容忍性（Partition Tolerance）这三项中的两项[2]。

FLP 不可能性可以采用更弱的终止条件或者更强的网络假设解决，Ben-Or 在 1983 年提出了完全异步模型下概率终止的共识算法[3]，Dwork 等人在 1988 年提出了部分异步模型下的共识算法，保证在网络同步的时侯算法能够确定性终止[4]，Leslie 等在 1982 年提出了同步模型下的共识问题[5]。

早期的共识算法一般也称为分布式一致性算法，与目前主流的区块链共识算法相比，分布式一致性算法主要面向分布式数据库操作、且大多不考虑拜占庭容错问题，即假设系统节点只发生宕机和网络故障等非人为问题（CFT, Crash Fault Tolerance），而不考虑恶意节点篡改数据等问题（BFT, Byzantine Fault Tolerance, 拜占庭容错）。拜占庭容错问题是 Leslie 在 1982 年提出的分布式领域容错问题，它是分布式领域中最复杂、最严格的容错模型[5]。在该模型下，系统不会对集群中的节点做任何的限制，节点可以向其他节点发送随机数据、错误数据，也可以选择不响应其他节点的请求.这些无法预测的行为使得容错这一问题变得更加复杂。

1.2 CFT 类共识算法

CFT 类共识算法只保证分布式系统中节点发生宕机错误时整个分布式系统的可靠性，而当系统中节点违反共识协议的时候无法保障分布式系统的可靠性，因此 CFT 共识算法目前主要应用在企业内部的封闭式分布式系统中。目前流行的 CFT 共识算法主要有 Paxos 算法及其衍生的 Raft 共识算法。此外，获得较多研究关注的共识算法还有两阶段提交（Two-Phase Commit） 算法、三阶段提交（Three-Phase Commit）算法、Zab、Kafka 等。

1.2.1 Paxos 共识算法

Paxos 共识算法是基于消息传递的一致性算法，由 Leslie Lamport 在 1990 年提出[6]，该算法基于同步性假设，通过引入超时（Timeout）概念规避了 FLP 不可能问题，如果在确认下个值的过程没有进展，Paxos 会等到超时，然后重新进行共识的步骤。Paxos 的算法步骤如下：

阶段 1：准备请求

1) 提案者选择新的提案号（n），然后向接收者发送“准备请求”。
2) 当接收者收到准备请求 (“prepare,” n)，且 n 比已经接收到的其他准备请求大，接收者会发出消息 (“ack,” n, n’, v’) 或 (“ack,” n, ^, ^)。
3) 接收者承诺不会再接纳任何提案号小于 n 的准备请求。
4) 如果有的话，接收者会发出具有最高提案号的提案值（v），反之则发出空消息 ^ 。

阶段 2：接收请求

1) 如果提案者收到大多数接收者的响应，则可以发出带有提案号 n 和价值 v 的接收请求 (“accept,” n, v) 。
2) n 是准备请求中出现的数字。
3) v 是响应中具有最高提案号的提案值。
4) 当收到接收请求(“accept,” n, v)时，除非接收者已经采纳了大于 n 的提案号，否则接收者就会采纳这个请求。

阶段 3：学习阶段

1) 每当接收者采纳了一个提案，就会向所有学习者返回 (“accept,” n, v)。
2) 学习者从多数接收者那儿得到 (“accept,” n, v)，选定最终的值 v，然后向其他学习者发送 (“decide,” v)。
3) 学习者收到 (“decide,” v) 和最终决定的值 v。

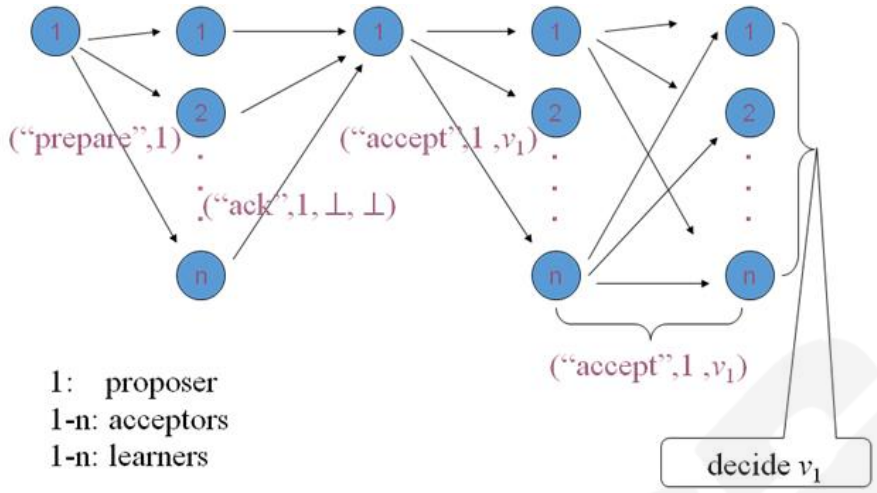


图 1Paxos 算法示意

为了提供部署的灵活性，Paxos 描述的许多主要特性都是开放式的，比如领导者选举、侦错、日志管理等内容。这种设计选择后来成为 Paxos 最大的缺点之一，使得 Paxos 不仅难以理解，而且难以实施。

1.2.2 RAFT 共识算法

因为 Paxos 晦涩难懂，2013 年 Ongaro 和 Ousterhout 为 Raft 复制状态机提出了一种新型共识算法[7]，其核心目标是可理解性，主要有两方面的改进：

1）将整体算法的描述分解为子问题

Raft 算法将其整体划分成了几个子问题，每个子问题都可以独立解释，从而理解 Raft 算法只需要相对独立地理解几个子问题即可。Raft 算法可分为以下几个子问题：

- **Leader election**：描述如何从集群节点中选举出 Leader；
- **Log Replication**：描述如何将日志同步到各个节点从而达成一致；
- **Safety**：定义了一组约束条件保证 Raft 算法的强一致性；
- **Membership changes**：描述如何变更集群关系（增加或者减少节点）。

2）简化状态空间

Raft 算法尽可能地确定各个环节的状态。典型地，Raft 算法采用 strong leader 模型，每个日志的读写均由 Leader 从中主动协调，整体系统的数据流变得非常简单：从 Leader 流向 Follower。而且每个节点的状态也只有 3 种：Leader，Candidate 和 Follower。如图 2 所示：

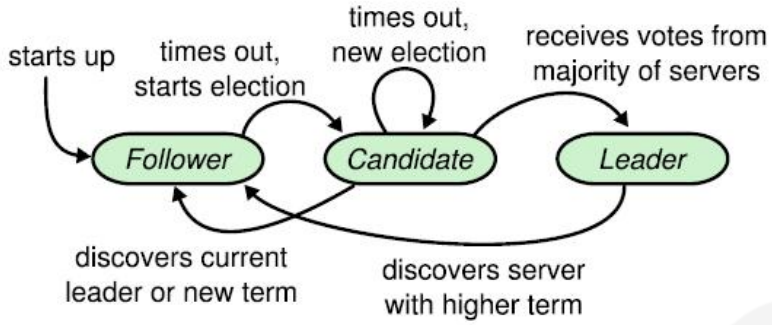


图 2 RAFT 数据流

此外， Raft 使用两个超时机制控制领导人选举，分别是选举超时及心跳超时。在 Raft 中，如果节点宕机并重启后，在试图声明成为领导者（Leader）之前，需要至少等待一个超时周期，并且一定会有所进展。

1.3 经典拜占庭共识

经典拜占庭共识算法能够保证当系统中不超过一点数量的节点发送拜占庭行为时，系统仍能够保持一致性。经典拜占庭容错系统普遍采用的假设条件包括：（1） 拜占庭节点的行为可以是任意的，拜占庭节点之间可以共谋；（2） 节点之间的错误是不相关的；（3）节点之间通过异步网络连接（如 Internet 网络），网络中的消息可能丢失、乱序到达、延时到达； （4）服务器之间传递的信息，第三方可以知晓但不能篡改或伪造信息的内容，可以通过签名技术实现这一点。

目前主要有两种类型的经典拜占庭协议，一种是基于状态复制的，称为 Agreement-Based 协议，通过节点间的相互通信达成对请求序列的共识；另一种是 Quorum-based 协议，客户端直接与节点交互，乐观地执行请求。前者一般通信复杂度较高，后者处理争议的代价比较高。

BFT 类算法效率一般较低，很难实用，学术界和业界对 BFT 协议进行了大量改进，其优化的主要思路是针对无拜占庭错误的场景，尽可能降低协议复杂度，提高效率。

1.3.1 Agreement-based BFT 协议

Agreement-based BFT 协议通常需要有一个主节点作为网络中的枢轴，与其他节点相比，主节点在共识过程中起最主要的作用，但通常也会成为系统性能的瓶颈。主节点需要将客户端发来的请求排序后发送给所有的备份节点，所有节点通过互相通信后达成一致，

实现安全性（Safety），大多数协议中所有节点也会向客户端回复响应，实现活性（Liveness）。该类协议通常需要 3f+1 个节点实现对 f 个拜占庭节点的容错。

1.3.1.1 PBFT 协议

PBFT 是 Castro 和 Liskov 在 1999 年提出的[8]，是第一个“实用”的拜占庭容错算法，解决了原始拜占庭容错算法效率不高的问题，将算法复杂度由指数级降低到多项式级（$O(n^2)$），使得拜占庭容错算法在实际系统应用中变得可行。首次提出在异步网络环境下使用状态机副本复制协议，并且通过优化在早期算法的基础上把响应性能提升了一个数量级以上。PBFT 中的很多机制都受到 Paxos 的影响。

PBFT 在保证活性和安全性（Liveness & Safety）的前提下，提供了(n-1)/3 的容错性，即如果系统内有 n 个节点，那么系统最多能容忍的恶意/故障节点为(n-1)/3 个。

PBFT 是一种状态机副本复制算法，所有的副本在一个视图（view）轮换的过程中操作，主节点通过视图编号以及节点数集合确定，即：主节点 p = v mod |R|。v：视图编号，|R|节点个数，p：主节点编号。算法主体流程如图 3 所示，共分为 5 个阶段：

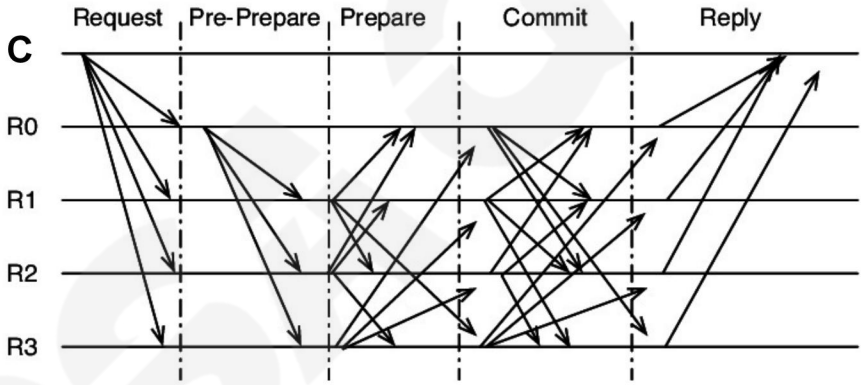


图 3 PBFT 算法流程

1. REQUEST

客户端 c 向主节点 p 发送<REQUEST, o, t, c>请求并签名，其中 o 为请求的具体操作，t 为时间戳，c 为客户端标识，REQUEST 包含消息内容 m，以及消息摘要 d(m)。

2. PRE-PREPARE

主节点对收到的请求做校验，如果校验通过，则分配一个编号 n，编号 n 主要用于对客户端的请求排序，然后主节点广播一条<<PRE-PREPARE, v, n, d>, m>消息给其他副本节点并签名，其中 v 为视图编号，d 为客户端消息摘要，m 为消息内容。

3. PREPARE

副本节点 i 收到主节点的 PRE-PREPARE 消息，进行校验，如果校验通过，则向其他节点包括主节点发送一条<PREPARE, v, n, d, i>消息并签名，其中 v, n, d, m 与上述 PRE-PREPARE 消息内容相同，i 是当前副本节点编号。记录 PRE-PREPARE 和 PREPARE 消息到 log 中，用于 View Change 过程中恢复未完成的请求操作。

4. COMMIT

主节点和副本节点收到 PREPARE 消息，需要进行校验。如果副本节点 i 收到了 2f+1 个验证通过的 PREPARE 消息，则向其他节点包括主节点发送一条<COMMIT, v, n, d, i>消息并签名，其中 v, n, d, i 与上述 PREPARE 消息内容相同。记录 COMMIT 消息到日志中，用于 View Change 过程中恢复未完成的请求操作。记录其他副本节点发送的 PREPARE 消息到 log 中。

5. REPLY

主节点和副本节点收到 COMMIT 消息，需要进行校验。如果副本节点 i 收到了 2f+1 个验证通过的 COMMIT 消息，说明当前网络中的大部分节点已经达成共识，运行客户端的请求操作 o，并返回<REPLY, v, t, c, i, r>给客户端，其中 r 是请求操作结果，客户端如果收到 f+1 个相同的 REPLY 消息，说明客户端发起的请求已经达成全网共识，否则客户端需要判断是否重新发送请求给主节点。记录其他副本节点发送的 COMMIT 消息到 log 中。

如果主节点作恶，它可能会给不同的请求编上相同的序号、或者不去分配序号、或者让相邻的序号不连续，副本节点应当有职责主动检查这些序号的合法性。客户端设置超时机制。如果主节点掉线或者由于作恶而不广播客户端的请求，一旦超时，则向所有副本节点广播请求消息。副本节点检测出主节点作恶或者下线，发起 View Change 协议。

为了确保在 View Change 的过程中，能够恢复先前的请求，每一个副本节点都记录一些消息到本地的 log 中，当执行请求后副本节点需要把之前该请求的记录消息清除掉。最简单的做法是在 Reply 消息后，再执行一次当前状态的共识同步，但这样做的成本比较高，因此可以在执行完多条请求 K（如 100 条）后执行一次状态同步。这个状态同步消息就是 CheckPoint 消息。

PBFT 算法由于每个副本节点都需要和其他节点进行 P2P 的共识同步，因此随着节点的增多，性能下降得很快，但在较少节点的情况下可以有不错的性能。

1.3.1.2 Chain 协议

Chain 协议是 Van Renesse 等在 2004 年提出的一个共识协议[9]，与 PBFT 不同的是其采用链式传播路径：从主节点开始，每一次传播时即加入该节点的摘要信息。当客户端较多时，Chain 通常会比 PBFT、Zyzzyva 等有更高的吞吐量。

1.3.1.3 Ring 协议

Ring 协议是 Guerraoui 等在 2010 年提出的共识协议[10]。该协议使用环形拓扑方式传递消息，即每个节点都有消息的上一个发送者和下一个接收者，以此方式降低对部分节点分配更多工作而形成的性能瓶颈问题。对有无错误的不同场景，Ring 分别采用两种运行模式：快速模式（Fast Mode）和弹性模式（Resilient Mode），并采用 ABSTRACT 框架进行切换。

1.3.1.4 Honey Badger BFT

Honey Badger BFT 是 Miller 等人在 2016 年提出的共识协议[61]。该协议基于异步网络假设，是第一个实用的异步 BFT 共识算法。相比于 PBFT，Honey Badger BFT 在现实的网络环境下具有更好的吞吐量及确认延迟。

Honey Badger BFT 并不分主节点和从节点，每个节点都公平的接收交易，每个节点都各自维护交易池。每轮开始时，每个节点会从交易池里随机选择出交易组成交易集合 U_i，并利用 Reliable broadcast（RBC）协议广播给其它节点。RBC 协议是为了降低广播带宽而设计的，其主要思路是发送节点将消息分割成 n 份，分别发送给其它的 n 个节点，n 个节点直接通过相互广播即可恢复出消息，其中为了能够容纳 f 个恶意节点，只需要其中 n-f 个碎片即可恢复出消息。接着节点运行二元拜占庭协议（Binary Byzantine Agreement，BBA）剔除无效交易和重复交易，得到一个异步公共交易子集（Asynchronous Common Subset）。

受 FLP 不可能定理所限，Honey Badger BFT 协议的在异步网络下为一个非确定性共识协议，可能无法终止。

1.3.1.5 Hotstuff

Hotstuff 是 Yin 等人在 2019 年提出的共识协议[62]。HotStuff 是一个三阶段投票的 BFT 类共识协议，通过引入门限签名及新型网络拓扑结构实现了 O(n)的通信复杂度，并通过流水线技术提高了共识效率。

Hotstuff将视图切换流程和正常流程进行合并，不再有单独的视图切换流程，降低了视图切换的复杂度。为此，Hotstuff为正常流程引入了新的确认阶段，把PBFT的两阶段确认扩展成了三阶段确认：pre-commit阶段、commit阶段及decide阶段。

为了减少通信量，HotStfuff引入了门限签名技术。在HotStuff的三阶段确认中，从节点会对消息进行门限签名并发送给主节点，当主节点收到足够多的签名时，将签名聚合导出完整签名，并在向从节点广播下一阶段消息时附上该签名，供从节点验证。

Hotstuff用门限签名和链式结构把节点轮换、区块链出块以及拜占庭共识结合了起来，使得无论是出块、拜占庭共识还是区块轮换都有了一个非常简洁并且完全一致的形式，并且将节点轮换和拜占庭共识的通信复杂度降到了O(N)。但缺点是由于多了一轮共识，其交易确认延迟相比其它BFT算法要高。

1.3.1.6 LibraBFT

作为Facebook DIME项目（曾经命名为Libra）的共识算法，LibraBFT在HotStuff的基础上引 入显示活跃度的机制并提供了具体的延时分析。

LibraBFT在3f+1个验证节点之间收集投票， 这些验证者可能是诚实的节点也可能是拜占庭节点。在网络中有2f+1个诚实节点的前提下， Libra能够抵御f个验证节点的双花攻击和分叉攻击。

LibraBFT在一个有全局统一时间 （GST），并且网络最大延时（ΔT）可控的Partial Synchrony的网络中是有效的。并且， LibraBFT在所有验证节点都重启的情况下，也能够保证网络的一致性。

严格说来，LibraBFT是基于HotStuff的一个变种，叫链式HotStuff（Chained HotStuff）。链式HotStuff是在基本HotStuff（Basic HotStuff）上引入流水线概念，进一步提升效率的一个改进共识协议。libraBFT最初会选择一些在不同地理上分布的创始成员做共识节点，以后逐渐的将共识节点对外开放，并基于libra稳定币的多少选择共识节点，也就是转变成PoS机制。

libraBFT的共识流程是分为不同轮次（Round），每一轮中一个Leader主节点被选 出。主节点会提议一个区块，里面包括多个交易。该区块将广播给其它共识节点。其它共识 节点会验证区块里的交易，并对其投票。主节点收到大多数（超过2f+1，f是系统中能容忍的拜占庭节点数）节点的投票后，主节点把确认消息发给所有共识节点确认。如果主节点没收到大多数投票，或者主节点出现故障，副本共识节点的定时将超时，副本节点会发起新一轮提议。

libraBFT 在 HotStuff 基础上的改进主要在于提供一个详细的参与同步轮次的 Pacemaker 设计和实现。并提供对实际交易确认的活性分析。LibraBFT 提供对共识节点投票权力的重配置机制。同时它给出了对提议节点和投票节点激励的机制。LibraBFT 的白皮书给出了如何检测投票节点破坏正确性的行为，为今后在协议中加入惩罚机制打下基础。同时白皮书也详细讨论如何同步，使得投票节点能同步它们的状态。libraBFT 白皮书采用 Rust 语言描述协议。在 LibraBFT 中，为了更好地支持 Libra 生态系统的目标，LibraBFT 以多种方式扩展 和调整了核心 HotStuff 协议和实现。重要的是，LibraBFT 重新定义了安全条件，并提供了 安全、活性和更高响应度的扩展证明。LibraBFT 还实现了一些附加功能。

首先，通过让验证器对块的结果状态(而不仅仅是交易序列)进行集体签名，LibraBFT 使协议更能抵抗非确 定性错误，还允许客户端使用法定人数证书来验证读取的数据库。

其次，LibraBFT 设计了 一个发出明确超时的起搏器，验证器依靠法定人数进入下一轮而不需要同步时钟。

第三， LibraBFT 打算设计一个不可预测的领导者选举机制，其中一轮的领导者由最新提交的块的提议者使用可验证的随机函数 VRF 确定。这种机制限制了攻击者可以针对领导者发起有效拒绝服务攻击的时间窗口。

第四，LibraBFT 使用聚合签名保留签署仲裁证书的验证者的 身份。这使我们能够为有助于仲裁证书的验证人提供激励，聚合签名也不需要复杂的密钥阈值设置。

具体实现细节如下：

LibraBFT 共识组件最主要的是实现了 Actor 程序模型，使用消息传递在不同的子 组件之间进行通信，其中 tokio 框架用作任务运行时。Actor 模型的主要例外是(因为它是由几个子组件并行访问的)是共识数据结构 BlockStore。BlockStore 管理块、执行、仲裁证书和其他共享数据结构。共识组件中的主要子组件是：

TxnManager 是内存池组件的接口，支持拉取交易以及删除已提交的交易。提议者使用来自内存池中的按需拉取交易形成提议块。

StateComputer 是访问执行组件的接口。它可以执行块，提交块，并可以同步状态。

BlockStore 维护提议块树、块执行、投票、仲裁证书和持久存储。它负责维护这些数据结构组合的一致性，并且可以由其他子组件同时访问。

EventProcessor 负责处理各个事件，如 process_new_round、process_proposal、process_vote。它公开每个事件类型的异步处理函数和驱动协议。

Pacemaker 负责共识协议的活性。它由于超时证书或仲裁证书而改变轮次，并在它是当前轮次的提议者时提出阻止。

SafetyRules 负责共识协议的安全性。它处理仲裁证书和分类信息以了解新的提交，并保证遵循两个投票规则 — 即使在重启的情况下（因为所有安全数据都持久保存到本地存储）。

所有共识消息都由其创建者签名，并由其接收者验证。消息验证发生在离网络层最近的地方， 以避免无效或不必要的数据进入协商一致协议。

1.3.2 Quorum-based BFT 协议

Quorum 机制是一种分布式系统中常用的机制，用来保证数据冗余和最终一致性的投票算法。其主要思想来源于抽屉原理，常用于分布式系统的读写访问控制[11]。

该类共识协议不需要节点间相互通讯，而是由节点直接执行并响应客户端发来的请求。当受到足够数量的响应后，客户端才会将结果最终提交。但是当出现拜占庭错误场景时，通常会花费较大的代价解决。另外，由于缺少对请求的排序机制，Quorum 方法无法处理有竞争（Contention）的情况。

1.3.2.1 Q/U 协议

Q/U 协议（Query/Update）是 Abd-El-Malek 在 2005 年提出的一个基于 Quorum 的协议[12]，该协议没有主节点为请求排序，而是由客户端直接向节点发送请求并由节点反馈结果，最多能容忍(n-1)/5 个恶意/故障节点。

1.3.2.2 HQ 协议

HQ 协议（Hybrid Quorum）Cowling 等人在 2006 年提出的共识协议[13]，该协议综合参考并优化了 Q/U 协议和 PBFT 协议，最多能容忍(n-1)/3 个恶意/故障节点。对没有竞争的情况，该协议对 PBFT 节点间通讯做了简化，将共识分为两个阶段：第一阶段是客户端发送请求并收集节点的状态信息；当收到结果表明 2f+1 个节点状态相同且可以执行请求后，开始第二阶段信息发送并由节点执行请求。如果发现有竞争，则采用类似 PBFT 的解决过程，性能退化为和 PBFT 类似。

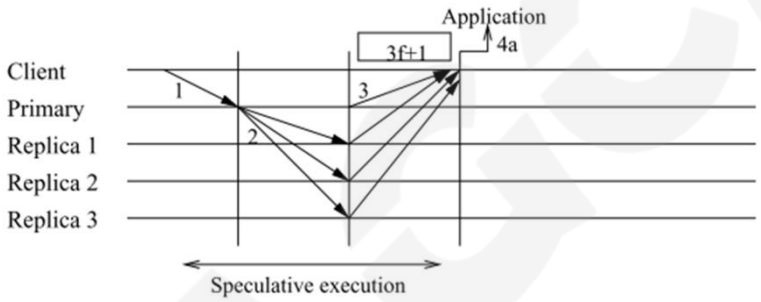
1.3.2.3 Quorum 协议

Quorum 协议是 Guerraoui 等人在 2010 年提出的基于 Quorum 机制的一个共识协议[14]，主要为没有客户端竞争的非异步网络设计，只需要 3f+1 个节点进行拜占庭容错。当没有错误产生时，Quorum 协议的传播路径和 Q/U 类似，节点独立执行请求并自己维护一个执

行历史记录。当客户端数量较少时，其吞吐量和延迟等性能指标均比其他 BFT 类共识更好。但其缺点也和 Q/U 类似，无法处理客户端有竞争的情况。

1.3.2.4 Zyzzyva 协议

Zyzzyva 是 Kotla 等人在 2007 年提出的一个协议[15]，需要 3f+1 个节点对 f 个恶意节点进行拜占庭容错。Zyzzyva 同样需要主节点将客户端发来的请求排序后转发给副本节点，但采用了更加乐观的策略，节点不需要通过需要消耗大量系统代价的 3 阶段提交过程即可响应客户端的请求，节点同意由主节点发出的排序请求即给客户端返回结果。由客户端而不是节点来负责考虑一致性问题，客户端会根据节点返回的一致性结果数量分别执行不同的动作。在乐观情况下，该协议具有线性的复杂度，但如果发生错误，主节点的更替仍然需要$O(n^3)$的时间复杂度。



Abraham 等人在 2017 年指出 Zyzzyva 协议存在安全漏洞[16]，并在 2018 年给出了修正方案[17]。

1.3.2.5 Zeno 协议

在 Zyzzyva 协议的基础上，Singh 在 2009 年提出的一个新的共识协议 Zeno[18]，该协议将 Zyzzyva 原有的强一致性替换为一个较弱的最终一致性保证，它允许客户端偶尔忽略其他更新，但是当网络不稳定时，所有节点的状态需要进行合并以达成一致。

1.3.3 基于拜占庭锁的共识协议

拜占庭锁是拜占庭协议的扩展，通过利用 IO 绝大多数时间里不会出现竞争的特性达到降低服务器响应时间、提高吞吐量与扩展性的效果。

Hendricks 等在 2010 提出的 Zzyzx 协议最早引入的拜占庭锁的概念[19]，协议包括加锁和解锁两个部分。加解锁过程均基于现有拜占庭协议达成对客户端授权的一致。当授权完成，获得锁的客户端可直接进行操作，去掉了主节点排序、节点间通讯等操作，从大幅度提高了吞吐量。但当有多个客户端需要频繁切换时，其性能也会大幅下降，因此该协议较为适用于客户端不会频繁发生变化的情况下。

针对无拜占庭错误的场景优化的共识协议，当遇到拜占庭错误时，其性能一般较低，甚至很难保证系统活性。为了解决这个问题，Aardvark 协议[20]、Prime 协议[21]、Spinning 协议[22]、RBFT 协议[23]等针对发生特定的拜占庭行为的场景进行了优化，降低了系统在有无拜占庭错误这两种场景下的表现差异。

1.3.4 其它共识协议

此外，Pass 等人将可验证随机数与中本聪共识相结合，提出了 Sleepy 协议[24]，该协议大大降低了共识协议的复杂性，并将通信复杂度降低到线性，但是该协议的确认时间较长，且无最终确定性。Baird 将 BFT 共识协议与 DAG 结合，提出了 Hashgraph 协议[25]，将整个共识过程融入到 gossip 通信中，降低了通信复杂度。

1.4 开放 BFT 类共识

传统的共识算法一般都假设网络是有准入机制的，因此无法应用于开放网络，2008 年中本聪在发表的比特币白皮书引入了基于工作量证明（Proof of Work, PoW）的共识算法[26]，为解决开放网络中的分布式一致性问题提供了新的思路，使得协调复杂网络环境中的成千上万节点成为可能。此后，受比特币启发，诞生了一大批适用于开放网络的共识算法，如权益证明（Proof of Stake, PoS）、空间证明（Proof of Space, PoSpace）、混合共识（Hybrid Consensus）等。

1.4.1 PoW 类共识

工作量证明的思想最早由 Dwork 等人在 1993 年提出，主要用来解决垃圾邮件的问题[27]。1999 年 Jakobsson 等人正式提出了“工作量证明”的概念，Adam Back 在 1997 年提出并在 2002 年正式发表了基于工作量证明机制的 HashCash，用以解决垃圾邮件问题[28]，与 Dwork 的工作量证明所不同的是，HashCash 的工作量证明机制更具随机性。Dwork 的工作量证明需要解决一个难题，这意味着一台速度更快的电脑总能比一台速度慢的电脑更快解决问题，而 Hashcash 允许速度较慢的计算机在某些时候更快地找到正确答案，仅在统计学上速度快的电脑能够更快的找到答案（这一点对 PoW 共识非常重要）。HashCash 为比特币的提出奠定了基石。

1.4.1.1 中本聪共识

中本聪通过引入经济激励及工作量证明保证比特币网络分布式记账的一致性，其核心思想是通过分布式节点的算力竞争保证数据的一致性和共识的安全性。比特币系统的各节点基于各自的计算机算力相互竞争共同解决一个求解困难但是验证容易的数学难题（求

Hash 的原象），最快解决该难题的节点将获得下一区块的记账权和挖块奖励。共识流程如下：

1) 客户端生成交易并向全网进行广播；
2) 比特币矿工节点将收到的交易放到交易池中，交易池中的交易会按照一定的优先级顺序打包进区块；
3) 矿工节点不断尝试为自己的区块找到一个具有足够难度的工作量证明；
4) 当矿工节点找到了一个工作量证明，就向全网进行广播该区块；
5) 其它节点对收到的块进行验证：打包的交易是否合法、工作量证明是否有效、引用的父块是否有效、时间戳是否有效等；
6) 若区块有效且以该区块结尾的链具有更大累积难道的工作量证明，则接受该区块并在该区块后继续出块。

中本聪共识基于同步网络假设，比特币出块间隔维持在 10 分钟左右，这就保证了区块能够在下一个区块产生前传播到几乎所有节点。当拜占庭节点的算力不超过 50%时中本聪共识能够保证系统的安全性。相比于传统共识算法，中本聪共识无法达成最终一致性，但由于每个区块都引用了上一个区块，因此每个区块都是对之前所有区块的确认，随着确认区块的增加，区块被修改的可能性也不断降低，从而实现统计意义上的最终一致性。

Garay 等人在 2015 年对比特币共识协议进行了理论分析，提出了 Bitcoin Backbone Protocol[44]，从理论层面解决了在只有随机出块节点竞选而没有显式的 BFT 委员会投票机制情况下，Bitcoin 的“最长链法则（the longest-chain-rule）”的有效性。Bitcoin Backbone Protocol 最早给出了在半同步前提下，线性表结构区块链的三个基本的共识特征：

1) Common Prefix：共有前缀，任意一对网络中诚实节点移除本地链（view）上最多 k 个最新块以后，得到的剩余区块相同，对应 BFT 协议里面 Agreement 特性要求；
2) Chain Quality：区块链质量，在链上任意 k 个连续块里，由攻击者生成的块的比例有上界，对应 BFT 协议里 validity 特性要求；
3) Chain Growth：区块链单向增长率，任意诚实节点在先后两个 time slot 之间观察到链上生成的区块数有下界，对应 BFT 的 liveness 特性要求。

1.4.1.2 GHOST

为了保证安全性，比特币将出块间隔定为 10 分钟，从而导致其低吞吐量（<10Tps）和长确认时延，通过增加区块大小及减小出块间隔可以解决这个问题，但这将导致：（1）分叉率增高，导致安全性降低；（2）公平性被破坏，网络延迟低的节点更有可能获得奖

励；（3）攻击成本降低，由于诚实节点的算力有一部分浪费在分叉上，恶意节点可以使用低于 51%的算力完成攻击。

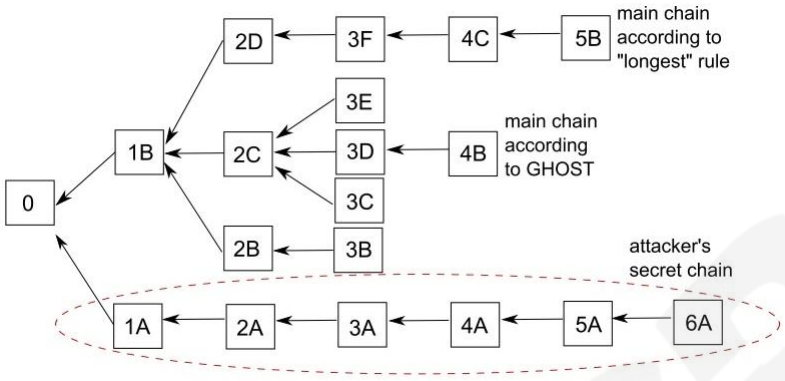


图 4 Gost 协议示意

为解决这个问题，Sompolinsky 提出了 GHOST 协议[30]，与中本聪共识协议相比，GHOS 协议修改了最长链规则，在每次分叉的时候选取拥有最重子树的分叉节点，如图 4 所示。以太坊的共识协议是 GHOST 协议的一个变体，其出块间隔做到了 15 秒左右。

1.4.1.3 Bitcoin-NG

Bitcoin-NG 是 Eyal 等人在 2016 年提出的一个共识协议，其目的是提高比特币交易的吞吐量[31]。Bitcoin-NG 将比特币出块分为了两个阶段，分别是 leader 选举及交易序列化，分别对应两种区块 Key 区块及 Micro 区块，如图 5 所示。Key 区块产生与比特币区块产生一样，平均间隔约 10 分钟生成一个，需要工作量证明。一旦 Key 区块被挖出，挖出该区块的矿工将负责生成后续的所有 Micro 区块，直到下一个 Key 区块被挖出。Micro 区块的生成不需要工作量证明，但为了防止生成的 Mircro 区块过多而阻塞网络造成 Key 区块分叉率增高，协议规定了最大的 Micro 区块出块速率。

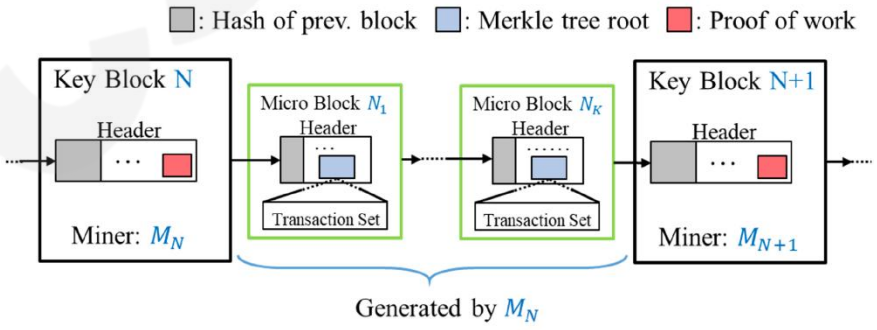


图 5 Bitcoin-NG 区块结构示意

所有的交易都被打包在 Micro 区块中，Micro 区块需要被下一个区块引用才有效，为了鼓励当前的 Key 区块引用上一个 Key 区块产生的 micro 区块，将 micro 区块的交易手续费的 60%发放给下一个区块，如图 6 所示。

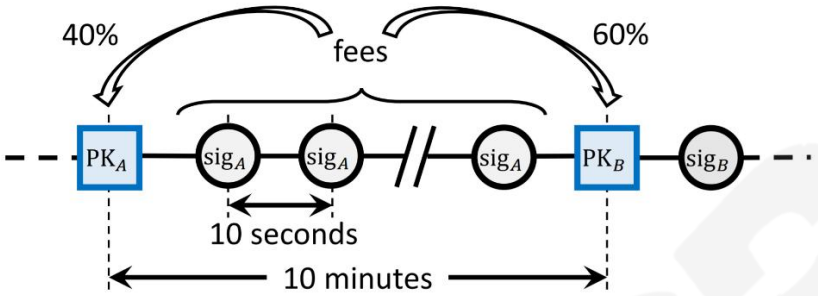


图 6 Bitcoin-NG 经济激励模型

Micro 区块的产生不需要工作量证明，因此节点能够快速廉价的产生微块， 从而恶意节点可以通过产生微块分叉发起双花攻击。为解决这个问题， Bitcoin-NG 引入了“毒药交易（Poison Transaction）”， 允许后面的出块者对恶意节点举报，举报成功恶意节点的酬金将被罚没，举报者获得恶意节点酬金的 5%作为奖励。

1.4.1.4 FruitChains

由于比特币挖矿难度较大，矿工为平衡收益易联合形成矿池，造成算力集中，并且利用自私挖矿，恶意节点可以通过创造分叉提高自己的挖矿收益，损害了系统的公平性。为了解决这个问题，Pass 等人在 2017 年提出了 FruitChains 协议[32]。

如图 7 所示，FruitChains 中将区块分成了两种，分别是正常区块及水果（Fruits），生成正常区块及水果都是通过寻找工作量证明完成，但是生成水果的难度要远低于生成正常区块的难度，由于水果与正常区块具有相似的区块结构及相同的工作量证明难题，因此水果与正常区块的挖矿可以同时进行。

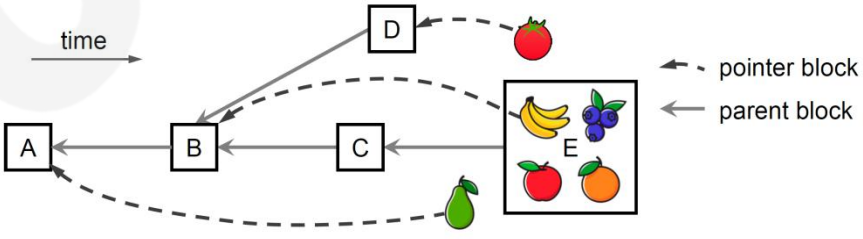


图 7 Fruitchains 结构示意

交易被打包在水果中，新挖出的水果只有被打包进正常区块才有效，如果攻击者通过藏匿区块进行自私挖矿或者扣块攻击，由于包含交易的水果总是会被其它诚实节点产生的区块打包，所有攻击会失败。隐私 FruitChains 能够有效地防止自私挖矿。

FruitsChain 中挖矿奖励及交易手续费会平均分发给所有的 Fruits，由于产生 Fruits 的难度要远低于产生区块的难度，矿工获得回报的频率将大大提高，从而不需要通过加入矿池来提高回报频率，避免了算力集中化。

1.4.1.5 Spectre

Spectre 是 Sompolinsky 等人在 2016 年提出的基于有向无环图（DAG）的共识协议[33]，其本质是利用区块之间的引用关系对交易顺序进行投票。不同于比特币，Spectre 没有主链的概念，区块可以同时引用多个之前的区块，所有的区块最终构成一个有向无环图（DAG）。与比特币最长链原则不同，当存在冲突区块时，Spectre 选择拥有最多子区块（投票）的区块，如图 8 所示。

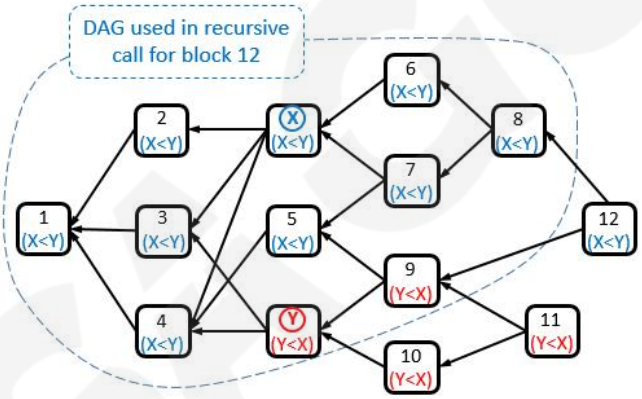


图 8 Spectre 结构示意

因为 Spectre 能够同时处理多个区块，其交易处理速度会明显提高，但是由于 Spectre 无法提高交易的全局序，仅能提供相对排序，因此其很难支持像智能合约这种对时许要求较高的功能。

1.4.1.6 Conflux

Conflux 是由 Chenxing Li 等人在 2018 年提出的基于 DAG 的共识协议[34]，该协议解决了基于 DAG 的共识算法无法提供全局序的问题。Conflux 采用延迟排序交易的方式，先乐观地并行执行交易（不论是否有依赖或者冲突）和出块（不论是否会产生分叉），然后经过一段时间再全局地对所有的交易排序，并删除那些有冲突的交易。

同 Spectre 不同的是，Conflux 采用了两种关系定义区块之间的关系：父边（Parent Edge）和引用边（Reference Edge），一个区块只能有一条指向父节点的父边，但可以有多条指向其它节点的引用边。其排序算法如下（如图 9 所示）：

1) 按照 GHOST 规则排序只包含父边的块，形成一个枢轴链（Pivot Chain）；
2) 根据枢轴链将区块划分成纪元（Epoch），然后对每个纪元里面的区块进行拓扑排序。确定 epoch 包含的区块的原则是：区块没有被之前的 epoch 包含且能够通过枢轴链的父边或者引用边遍历到；
3) 根据 happens-before 原则对不同 epoch 之间的块进行排序

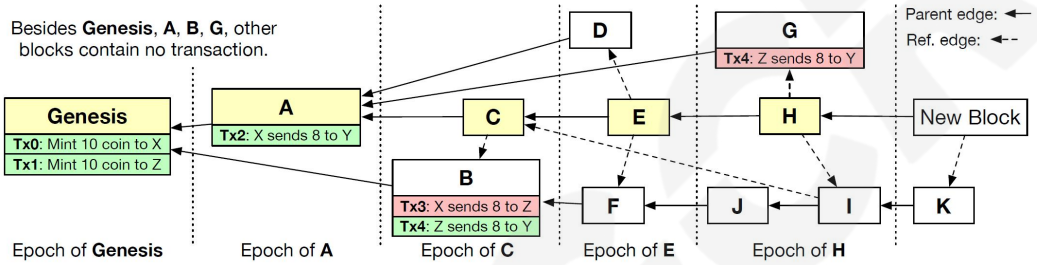


图 9 Conflux 结构示意

1.4.1.7 NC-MAX

Zhang Ren 等人对 PoW 共识中孤块的产生进行了分析[35]，将区块广播与新交易广播并行化，有效地缩短了区块广播时延，进而降低了孤块率。具体的，NC-Max 设计了一个交易两步确认机制，在致密区块中既包含新交易的哈希值（propose 交易），又包含老交易的哈希值（ confirm 交易）。Propose 交易的目的是通知其他矿工同步新交易，confirm 交易是将交易打包上链，confirm 交易必须是前几个块中已经通知过的新交易，留给矿工足够的时间对新交易进行同步。该机制实现了交易确认与交易同步的解耦，在同样的出块间隔下，孤块率得到显著的改善。

此外，区块链网络的状况并非一成不变，将出块间隔设置为固定时间并不能充分利用网络带宽。NC-Max 指出孤块率是区块链要稳定的目标，于是 NC-Max 改变了难度调整的策略，以孤块率作为难度调整的依据。当孤块率大于预设的安全阈值时，难度上升，区块间隔上升，从而使得孤块率回落到安全阈值，反之亦然，从而使得 NC-Max 的吞吐量能够达到安全阈值下网络所能承载的最大吞吐量。

1.4.2 PoS 类共识

PoW 共识算法需要消耗大量的能源，为了解决这个问题，一位名为 Quantum Mechanic 的数字货币爱好者在比特币论坛首次提出了基于权益证明（PoS）的共识算法[36]。

权益是指节点或用户拥有的资产（如代币），并根据用户拥有资产比例决定其成为下一个区块生产者的概率。拥有资产比例越高，其成为生产者的概率就越大。PoS 一定程度上解决了 PoW 算力浪费的问题，并为共识优化带来了新思路：通过代理人制度缩小共识范围，从而缩短达成共识的时间，提供系统的吞吐量。

1.4.2.1 Peercoin

Peercoin 是由 King 等人在 2012 年提出的，首次引入了权益证明（PoS）概念[37]。Peercoin 将区块分成了两种，分别是 PoW 区块与 PoS 区块。PoW 区块生成同比特币一样，需要工作量证明。PoS 区块的生成则需要消耗币龄（币的持有时间），节点拥有的币数量越多，拥有币的时间越长，其被选为出块者的概率越大。具体的，Peercoin 将随机散列运算的搜寻空间与币龄挂钩，限制了搜寻次数，避免了因大量计算而造成的能源消耗。

但 Peercoin 的币龄机制存在被攻击的可能，该机制无法充分激励节点始终保持在线状态，节点可以先保持离线状态，在积累一定的币龄之后才参与共识网络，然后再次离线。

1.4.2.2 Casper FFG

Casper FFG 是由 Buterin 和 Griffith 在 2017 年提出的基于 PoS 的共识协议[38]，其目的是为以太坊的 PoW 共识协议引入即时确定性（Instant Finality）。Casper FFG 许多设计受到 PBFT 的启发，并可以被视为改良后的 PBFT —它继承了 PBFT 的重要设计，同时添加了新的机制并简化了若干规则。

在 Casper FFG 中，区块产生仍然依靠以太坊的 Ethash 工作量证明算法，每隔一定高度（如 50）的区块被称为检查点（Checkpoints），检查点组成检查点树（Checkpoint Tree），验证者需要通过投票对检查点形成共识，投票的内容是由两个不同高度的检查点组成的连结（Link），其投票过程与 PBFT 类似。

节点可以通过抵押以太币成为验证者，节点被选中成为检查者的概率与抵押以太币的数量成正比，被选中的节点需要经过网络中现有验证者的共识才能最终成为验证者。为了使 Casper FFG 具有抵抗共谋的能力，系统会对验证者的违规行为做出惩罚，违规投票的验证者的押金会被罚没。由于验证节点集合会动态地随着时间变化，验证者的某些违规行为无法进行归责，为此 Casper FFG 引入了缝合机制（Stitching Mechanism），确保每个错误都能够被归责。

Casper FFG 更多的是作为一个过渡协议，使得以太坊能够从 PoW 成功过渡到 PoS，目前在研究中的还有 Casper CBC 版本，可以视为以太坊切换到 PoS 后的一个更好的版本。

1.4.2.3 Snow White

Snow White 是由 Bentov 等人在 2016 年提出的基于 PoS 的共识协议[39]，同比特币类似，Snow White 出块也是需要找到某个低于目标值（target）的 hash 值的原象。不同的是，Snow White 中节点只能每隔一段时间尝试一次寻找（通过将时间作为唯一随机源实现），此外 target 的大小与参与者抵押的押金成正比，从而实现参与者出块概率与抵押押金成正比。Snow White 采用与 Fruitchains[32]相同的区块结构机激励机制，挖矿收益交易放在水果中，水果放在区块中，将连续几个区块的奖励和其中包含的交易费平均分给水果的生产者，实现了公平性。

Snow White 基于弱同步假设，并采用睡眠模型（sleepy model）的网络模型，即假设节点不会保持永久在线，可能在某段时间内离线，又在某段时间内在线，间断的参与共识，该模型中节点的状态更加符合现实网络中节点的状态，因此适应性更广。

1.4.2.4 Ouroboros

Kiayias 等人在 2017 年提出的共识协议 Ouroboros [40]并对其安全性给出了形式化的证明，Ouroboros 是第一个可证明是安全和健壮的 POS 算法。Ouroboros 协议基于同步网络假设，其本质是通过多方运行 coin-tossing 协议生成随机数种子，用以随机选取出块节点及下一轮的出块节点候选人集合。

Ouroboros 将时间划分为一个个的 epoch，每个 epoch 又被划分为多个 slot，每个 slot 产生一个区块，如果产生的区块非法或者出块者不在线，则该 slot 会被跳过。Ouroboros 假设每个 epoch 内权益的分布是不变的，即权益是根据每个 epoch 开始前的历史计算，当前 epoch 权益分布的变化只能在以后的 epoch 中体现。在每个 epoch 的开始阶段会有一个该 epoch 的创世区块，记录了该 epoch 的出块节点候选人及随机种子ρ，但该创世区块并不上链。随机种子ρ是在上一个 epoch 中通过多方的 coin-tossing 协议生成的。Epoch 中的每个 slot 都会从该 epoch 出块节点候选人中随机选择一个出块节点，候选人被选中的概率与其权益的占比成正比，即$U_j = F(S_e, \rho, j)$，其中 S_e 为候选人集合，ρ为该 epoch 的随机种子，j 代表第 j 个 slot。

为了确保激励出块节点始终遵循该协议，Ouroboros 在每个 slot 除会选出一个单一的出块节点外，还会选出多个称为“背书节点”（endorsers）的验证节点。背书节点会对交易的正确性进行检查并为其背书，区块中所包含的交易必须均已由背书人背书的情况下才有效。此外为了激励除出块节点外的其它节点在线，Ouroboros 会按贡献度为权益持有者发放奖励。

但原始的 Ouroboros 协议存在问题：（1）由于伪随机函数是公开的，恶意节点可以在 epoch 开始时推测出整个 epoch 中所有的出块节点，从而进行针对性的贿赂或 DDoS 攻击；（2）用于计算随机种子的安全多方计算的计算协议性能会随参与节点的增多而降低；（3）其安全性论证基于同步网络模型假设。为了解决这些问题，David 等人在 2018 年提出了 Ouroboros Praos 协议[41]。

Ouroboros Praos 协议的主要改进是采用可验证随机函数（VRF）代替公开伪随机函数进行出块节点的选举。节点通过可验证随机函数 VRF 产生可验证随机数，如果其数值低于某个目标值， 则可确定被选中为出块节点，出块节点在生成区块时会附带产生的随机数和证明，网络中其他节点可以据此确认其合法性。因为每个节点随机函数是独立的，所以 Praos 并不能像原始 Ouroboros 那样保证每个 slot 都刚好有且只有一个出块节点，可能没有人选中，也可能选中多个。不过 praos 已经证明，在半同步网络模型上其依然可以保证原来的安全属性。同时 praos 的 slot 时长也比原始协议要低。

Badertscher 等人对 Ouroboros Praos 协议进行了改进，并参考了 Snow White 协议针对节点离线的情况的设计，提出了 Ouroboros Genesis 协议[42]，解决了困扰 PoS 的长链攻击问题（Long-range Attack），并基于 UC Framework 证明了其安全性。Ouroboros Genesis 保留了 Praos 协议里 VRF 的部分，考虑到 Praos 的 VRF 会像 PoW 一样可能在同一个 slot 里选出多于一个的合法出块节点而导致分叉，Ouroboros Genesis 修改了基本的最长链原则（Longest-chain-rule），新节点在加入网络时，需要从不同节点获得多个链，并进行对比，最终选定的区块链需要与其他链具有共同前缀且是最长链。这个新的设计保证新加入节点或者掉线节点能够在不引入需要附加信任机制的链上检查点（Checkpoint）的情况下正确自启（Bootstrap）。

Kerber 等人在 2019 年提出了 Ouroboros Crypsinous[43]，首次对支持隐私保护的 PoS 区块链协议进行了形式化分析，并引入 SNARKs 及 key-private forward secure encryption，使其能够更好的抵抗适应性攻击（Adaptive Attacks）。Ouroboros Crypsinous 与 Ouroboros Genesis 协议基本一致，区别主要在于出块节点的选取及交易处理。

1.4.3 PoSpace 类共识

空间证明（PoSpace）是为了解决 PoW 算法能源浪费提出的，节点需证明其在一定的磁盘空间中存储了特定的数据，节点被选中成为出块节点的概率与其所能证明的磁盘空间大小成正比。Burst 是首个基于空间证明的区块链项目。空间证明最大的问题是，因为挖

矿不需要付出代价，某个拥有大量磁盘资源的节点可以不费代价的覆盖整条链，从这条链最初的创世区块开始，把整个链重新挖一遍，独吞所有的区块奖励。

1.4.3.1 Permacoin

Permacoin 是 Miller 等人在 2014 年提出的共识协议[45]，Permacion 本质上仍是一个 PoW 共识协议，不过为了使挖矿变得有“价值”，Permacoin 会强制矿工存储一部分数据，并在出块时向其它节点证明一直存储着该数据，否则将无法出块。

1.4.3.2 Spacemint

Spacemint 是由 Park 等人在 2015 年提出的基于空间证明的共识协议[46]，Spacemint 详细分析了 PoSpace 协议所存在的两个问题 Grinding 和 mining multiple chains，并给出了解决方案。同 Permacoin 不同的是，Spacemint 存储的是无用数据，但是避免了因不断“搜寻”而引起的能源浪费。

Spacemint 的空间证明本质上是求解单向函数的逆，逆的求解很难在短时间内通过纯计算完成，但通过将函数的象及原象预先存储在磁盘上，通过查找磁盘就可以在极短的时间内找到答案，而找到答案的数量（或质量）与所用的磁盘空间成正比，从而实现了容量证明。

首先，Spacemint 划分了一段段固定的时间，每一段固定时间内只能出一个块，其出块过程为：每一段固定时间都有对应的挑战（challenge），矿工根据挑战在磁盘上搜寻对应的答案，找到答案的矿工会广播自己的答案并与其它矿工的答案进行对比，拥有最高质量的答案的矿工获得出块权。

由于 Spacemint 挖矿不需要代价，拥有大量磁盘空间的恶意节点可以私下构造一条具有更高质量（Chain Quality）的链而覆盖掉原有的链（Challenge-grinding Attack），为了解决这个问题，Spacemint 在计算 chain quality 时加入了衰减因子，使新的块占有更大的比重，从而增加了攻击难度。此外对应同时在分叉上出块，Spacemint 引入了惩罚机制，促使矿工在单一分叉上出块以加快收敛速度。

1.4.3.3 Chia

尽管 Spacemint 在计算 chain quality 时加入了衰减因子企图解决 challenge-grinding attacks，但仍然有被攻击的可能，为了解决这个问题，Bram Cohen 利用可验证延迟函数，将空间证明与时间证明结合，在其设计的 Chia 协议中，创造性地提出了时空证明机制（Proof of Space And Time）[47]。

可验证延迟函数（VDF）使一类数学函数，在允许同时使用少量 CPU 进行并行计算的情况下，能够使得该函数的计算需要至少一段已知的时间，验证者能够对函数的输出结果进行验证，一般验证时间要远低于计算时间。同 Spacemint 一样，矿工仍需要对 challenge 在其磁盘空间中搜寻答案，拥有最高质量答案的节点获得出块权，不同的是出块节点会将块广播给 VDF 服务器，VDF 服务器需要提供一段时间流逝的证明，以证明两相邻块之间间隔了足够的时间，系统中会有多个 VDF 服务器，最先完成 VDF 计算的服务器进行最终出块。

通过引入时间证明机制，Chia 解决了 challenge-grinding attack，因为攻击者如果想覆盖整条链的话，其不但要付出存储空间资源，还需要 CPU 的时间资源，而后者将使攻击很难进行。

1.4.3.4 Filecoin

Filecoin 是 Benet 等人在 2018 年提出基于容量证明的区块链协议[48]，其共识机制与 Chia、Spacemint 等共识机制最大的不同是，其磁盘中存储的是有用数据，避免了磁盘资源的浪费。

Filecoin 的共识机制是预期共识（Expected Consensus，EC），Filecoin 的矿工在每一轮里会独立的检查是否满足：

$$\mathcal{H}\left(\langle t||\mathsf{rand}(t)\rangle_{\mathcal{M}_i}\right)/2^L \leq \frac{p_i^t}{\Sigma_j p_j^t}$$

如果满足则获得出块权，其中 H 为 hash 函数，L 为函数值位数，<*t||rand(t)*>为第 t 轮的全网统一随机数，p_i^t为第 t 轮矿工 i 的存储空间，为啥出块节点的选取不可预测，$<m>_{M_i}$定义为：

$$\langle m\rangle_{\mathcal{M}_i} := \left((m), \mathsf{SIG}_{\mathcal{M}_i}\left(\mathcal{H}(m)\right)\right)$$

矿工获得出块权力的概率与矿工已拥有的有效存储空间占全网存储空间的比重成正比。因为计算相互独立，因此 Filecoin 会出现在 t 轮没有节点或有多个节点获得出块权的情况。此外，Filecoin 设有抵押机制，会强制矿工选择一条链，对同时挖多个链的矿工进行惩罚，有效促进矿工收敛。

为计算矿工算力及存储的有效性，保证 EC 共识机制的运行，Filecoin 协议中采用了复制证明（Proof-of-Replication）和时空证明（Proof-of-Space-time）两种算法，其中复制证明用来验证矿工已经存储了某文件，时空证明用证明矿工一直存储了该文件。

复制证明的目的是防范三种常见的攻击：女巫攻击，外源攻击和生成攻击，以保证矿工实际存储的数据大小与其声明存储的数据大小一致。时空证明可以理解为持续的复制证明，即矿工必须不断的生成证明，并在一个提交周期内提交存储证明。如果存储服务商没有在提交周期内连续及时提交证明，会被系统扣除部分代币。生成时空证明的过程跟复制证明非常相似，只是时空证明的输入是上一次生成的证明的输出，从而形成证明链，利用时间戳锚定这些证明链，这样即使验证者（Verifier）不在线，也能够在将来去验证矿工在该段时间内生成了证明链，PoSt 会被提交到链上用来产生新的数据区块。

1.4.4 proof of elapsed time

消逝时间证明是基于硬件芯片执行某个命令的等待时间实现的，其实质是利用可信硬件（TEE）产生随机数决定下一个区块生产者。在 Hyperledger Sawtooth 项目中，参与者使用英特尔可信芯片 Intel SGX 中的“飞地”模块，根据预定义的概率分布生成一个随机的等待时间，等待时间最短的节点被选为领导节点，SGX 可帮助节点创建区块、生成该等待时间的证明，而这种证明易于被其他 SGX 节点验证。Zhang 等人在 2017 年提出了资源高效利用挖矿（Resource-Efficient Mining，REM）的共识协议 PoUW[49]。在 PoUW 协议中，CPU 在承担原来的工作之外，还可以同时承担区块链的工作，该协议核心仍是采用可信硬件来计算 PoW，用时最短的节点将获得出块权。

1.4.5 混合共识机制：单一委员会

混合共识机制本质上是将共识范围缩小以提高共识效率。经典的分布式共识机制共识效率高、确认延时低，但由于存在女巫攻击，无法适用于开放的网络环境。混合共识机制将经典分布式共识机制与区块链共识机制相结合，即采用 PoW 或 PoS 的方式进行委员会选举解决女巫攻击，在委员会内部运行经典分布式共识机制，提高共识效率。

混合共识机制可分为单一委员会制度或者多委员会制度，采用单一委员会的混合共识机制选举一个委员会负责全网所有交易的处理，而采用多委员会的混合共识机制选举多个并行运作的委员会，将全网划分为多个片区，分片处理网络中的交易（扁平），或者对系统进行分层，由不同的层处理不同的逻辑（垂直）。 混合共识机制的一般过程如下：

1) 委员会成员选举。混合共识机制首先需要选举一个小型的委员会，为防止女巫攻击，委员会成员通过 PoW 或 PoS 的方式选举，节点被选中的概率与节点所拥有的算力或持有的权益成正比。
2) 委员会成员间共识。委员会成员间运行改进的 PBFT 协议或类似算法，实现委员会内部的拜占庭容错，进而产生区块。

3) 广播区块。委员会成员将生成的区块广播至全网，使得网络中非委员会的节点和客户端收到新产生的区块，更新交易和区块链。

4) 委员会重配置。为防止敌手腐化/DDos 委员会成员，在工作一定时期后，委员会成员应当更新，可以一次更新一个或多个成员，也可以全部更新。如果更新过慢，则会给敌手 DoS 攻击提供稳定有效的目标，但如果更新过快，更新委员会的成本更高且需要激励新一轮被选中者持续在线。

单一委员会的混合共识机制面临的安全问题主要是恶意节点干扰委员会选举和重配置过程。

1.4.5.1 PeerCensus

比特币共识机制较弱，只能提供最终一致性，导致其交易确认速度过慢。为了解决这个问题，Decker 等人在 2016 年提出了共识协议 PeerCensus [50]，首次将经典分布式一致性算法 PBFT 与 PoW 算法相结合，其主要想法是解耦区块创造过程和交易确认过程，以便共识速度可以显著提升。

如图 10 所示，PeerCensus 底层仍然是一般的区块链（如比特币），矿工挖到区块后将区块提交到上层的 Chain Agreement（CA），Chain Agreement 运行 PBFT 协议对区块确认，确认后的区块再返回给底层区块链。节点加入委员会需要提交工作量证明并经过委员会其它成员共识后才能加入。但 PeerCensus 委员会管理存在一定问题，它采用离开检测机制来判断节点是否离开。正常节点通过发送 ping 消息判断其它节点是否离线，如果离线则发起对该离线节点的“离开”提议，提议通过委员会共识后，委员会才完成重配置。这样的问题是，如果恶意用户不断制造“离开”提议，整个系统的运行效率将大大降低。

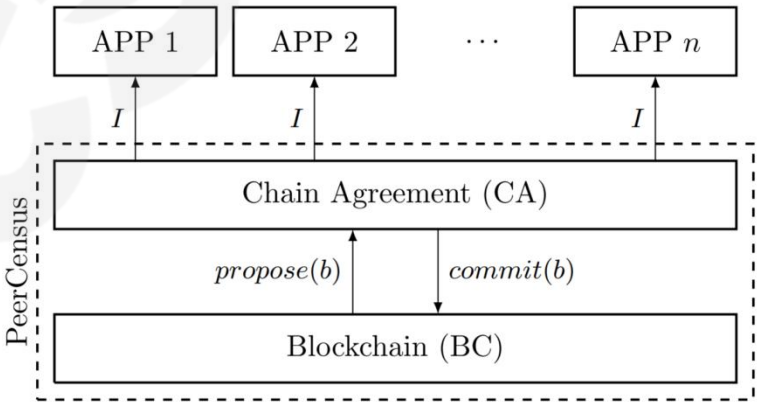


图 10 PeerCensus 结构示意

1.4.5.2 ByzCoin

ByzCoin 是 Kogias 等人在 2016 年提出的[51]，是首个能够支持比特币中工作量证明的基于动态成员关系的拜占庭共识协议。Byzcoin 将 PoW 与 PBFT 协议相结合实现了交易的即时确认并能够提供前向安全性。同时 Byzcoin 通过引入联合签名方案减小 PBFT 轮次的开销，使其能够支持更多的共识节点。

ByzCoin 借鉴了 Bitcoin-NG 的思想，将区块分为关键块（Keyblock）和微块（Microblock），关键块决定委员会成员和领导者。首先 ByzCoin 定义了一个窗口大小（144 个关键块或者 1008 个关键块），窗口会随着关键块的产生不断往前移动，处于窗口内的关键块的生产者组成委员会，成员的投票权由其产生的关键块的比例决定。同 Bitcoin-NG 一样，一旦一个节点生成了关键块，它将成为首领，被允许以固定速率生成微区块，微区块包含交易等信息。Byzcoin 架构如图 11 所示。

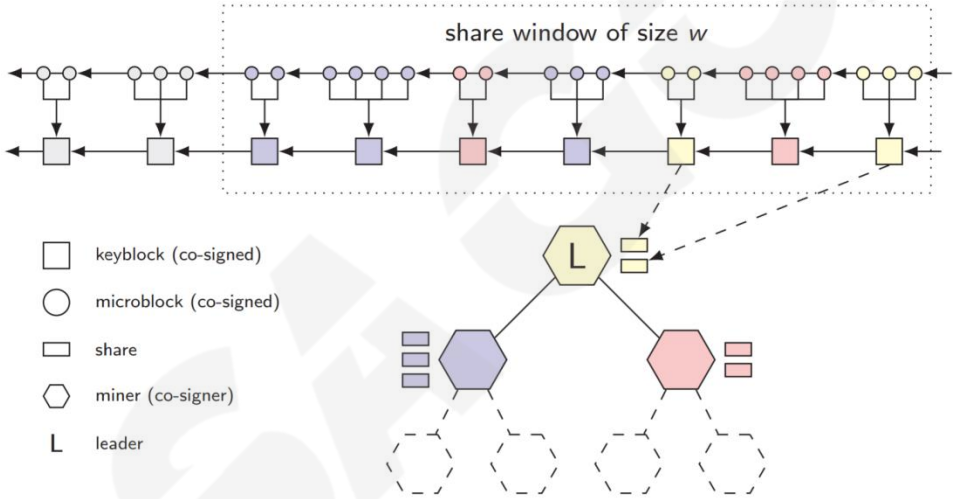


图 11 Byzcoin 架构示意

委员会成员采用改进的 PBFT 算法对微块进行共识，为了提高效率，ByzCoin 利用联合签名（Scalable Collective Signing）将 PBFT 的通信复杂度降到了 O(n)，使其能适用于规模较大的共识节点。此外为了激励委员会成员保持活跃，ByzCoin 会将部分挖矿奖励分发给参与共识的委员会成员。

1.4.5.3 Solida

Solida 是由 Abraham 等人在 2016 年提出的基于 PoW 和 BFT 共识的混合共识机制[52]。相比于 PeerCensus 及 ByzCoin，Solida 给出了更加详细的协议设计，尤其是委员会成员的重配置机制，解决了重配置时可能出现的问题。此外，Solida 严格证明了当恶意节点算力不超过 1/3 时，协议能够保证安全性及活性。

矿工可以通过求解 PoW 谜题加入委员会，一旦找到一个对应的 PoW，矿工可以向委员会成员广播该 PoW，委员会成员通过运行重配置协议，将该矿工加入到委员会中。Solida 的领导者选举方式借鉴了 Paxos 的思想，委员会每个成员都会被赋予一个等级元组 *(c, e, v)*，并依次按 c、e、v 进行排序，其中：

- c 代表委员会序号，每当新节点通过委员会重配置加入到委员会中后，委员会中节点的等级元组会由(c, e, v) 变为(c + 1, 0, 0)，新加入的节点成为委员会领导者；
- e 代表一个 PoW 对应的生存时期，如果有节点新找到 PoW，则其等级元组(c, e, v) 变为(c, e + 1, 0)，新找到 PoW 的节点成为领导者。
- v 的含义是视图，当现任领导者停滞不前，利用函数 L(c, e, v) = H(c, e) + v mod n 在委员会内部选出新的领导者，领导者元组由(c, e, v) 变为(c, e, v + 1)。

领导节点负责打包交易并创建区块，委员会成员通过运行改进的 PBFT 协议对区块进行共识，除此之外，委员会成员还需对重配置事件进行共识，以顺利更新委员会成员及领导节点。

1.4.5.4 Hybrid Consensus

混合共识 Hybrid Consensus 是由 Pass 等人在 2017 年提出的共识机制[53]，将经典共识与区块链共识结合，提高非授权网络下的交易确认速度。Hybrid Consensu 证明了在异步或者半异步假设下（网络传播延迟上限不可知），非授权网络下的共识无法保证一致性及活性。Hybrid Consensus 通过对系统进行更强的假设，通过混合共识机制实现了交易的快速响应，即交易的确认时间与网络真实时延有关，而与网络时延上限无关，协议的一致性及活性也被严格证明。

在 Hybird Consensus 中，底层链成为 snailchain，是一条 PoW 链，委员会的成员由最近出块的矿工组成，委员会成员之间运行 DailyBFT 协议对交易进行确认。

Hybrid Consensus 解决了 ByzCoin 中存在的因自私挖矿导致诚实节点维护的关键块链质量下降的问题，即在自私挖矿存在的情况下，要想诚实节点产生的关键块的比例超过 2/3，以使委员会中有足够多的诚实节点，那么敌手所占的算力不能超过 1/4。

1.4.5.5 Thunderella

Thunderella 是 Pass 等人在 2018 年提出的混合共识机制[54]，与其它混合机制不同的是，委员会成员之间并不运行完整的 BFT 类共识协议，如果要发生视图切换（View Change）及状态恢复，则通过底层的 PoW 协议实现。Thunderella 提出了乐观响应的概念，

即在乐观情况下，由上层的委员会快速处理交易，而一旦遭到攻击，则退回到底层 PoW 协议进行安全处理。

Thunderella 的委员会选举可以通过 PoW 进行也可以通过 PoS 进行。如果通过 PoW 进行，则最新找到 PoW 的 n 个节点成为委员会成员，并按顺序依此替换老节点。如果通过 PoS 的方式进行选举，节点需要现在底层链上抵押一部分保证金，系统随机从抵押保证金的节点中选择 n 个节点作为委员会成员，选举方法与 snow white 类似。

如果领导节点诚实且委员会中诚实节点数量超过 3/4，则系统处于乐观期，此时客户端将交易提交给领导节点，领导节点对交易进行排序并广播给其他委员会成员，委员会成员对交易进行签名，如果超过 3/4 个委员对交易进行了签名，则交易被最终确认。此外领导节点会定期的发送快速通道的哈希值给委员会成员确认，如果确认成功则该哈希值会作为心跳标记（Heartbeat）写入底层链，心跳机制保证了在回退时只会影响最后一个心跳标记后的交易。

如果不满足乐观条件则系统进入宽限期（如 Yell Transaction 长时间未被确认，心跳标记长时间未出现），此时委员会成员停止对来自领导节点的消息进行签名，但允许节点发布经过公证但没有包含在最近心跳标记中的交易到慢速链上，宽限期包含 k 个区块，宽限期结束后， 节点输出所有已确认和未确认交易，进入底层区块链确认期。

底层区块链确认期采用底层链共识机制，所有的交易处理和传统区块链没有区别。系统会等待足够的区块链数量后，重新进入乐观期，重启快速通道。

Thunderella 协议总体的安全性继承于底层链的安全性，底层链基于同步假设，实现了 1/2 的容错性，为部分同步假设下的上层 BFT 共识提供了 1/2 的容错能力。

1.4.5.6 Algorand

Algorand 是 Gilad 等人在 2017 年提出的基于同步假设的混合共识机制，Algorand 将 PoS 共识与经典分布式一致性算法相结合，并利用可验证随机数（VRF）进行委员会秘密选举，避免了被定向攻击的可能。

同经典的 PoS 共识一样，节点需要先抵押一定数量的代币才能参与到共识中。在每一轮共识开始时，每个符合条件的节点都可以通过 VRF 独立验证自己是否是潜在的领导节点或委员会成员，即如果$H\left(SIG_i\left(r,1,Q^{r-1}\right)\right)\leq 1/SIZE\left(PK^{r-k}\right)$则成为本轮的领导节点，如果满足$H\left(SIG_i\left(r,1,Q^{r-1}\right)\right)\leq n/SIZE\left(PK^{r-k}\right)$则成为本轮委员会成员，其中 H 为哈希函数，$SIG_i(_)$为签名函数，$Q^{r-1}$是第 r-1 轮的种子，$SIZE\left(PK^{r-k}\right)$为第 r-k 轮符合条件的公钥

数，k 为回溯系数。如果有多个候选 leader，最终上述哈希值最小的 leader 将在后续的共识中成为本轮最终的 leader。

委员会成员通过运行改进的分布式一致算法 BA*对区块进行共识。BA*包括分级共识协议（Graded Consensus，GC）和二元拜占庭一致协议（Binary Byzantine Agreement，BBA），Algorand 利用 GC 将对整个区块哈希值的共识转化为对二元数值的共识，并通过 BBA 对二元数值达成一致从而确认本轮区块。BBA 可以在诚实节点超过 2/3 的情况下快速达成共识，其具体过程是一个 3 步循环，循环中每一步都有 1/3 的概率达成共识。

尽管概率非常低，Algorand 仍然有可能发生分叉。如果发生分叉，Algorand 采用中本聪共识中的最长链法则进行恢复。

1.4.6 混合共识机制：多委员会

多委员会的混合共识机制是通过设立多个委员会，每个委员会负责不同的任务，以提高交易的处理能力，按委员会的拓扑结构不同，可分为分层结构（如 RSCoin，ELASTICO）和扁平结构（如 Chainspace，Omniledger）。相比于单委员会机制，多委员会机制在选举委员会成员后, 需要将新选举的委员会成员分配到不同委员会中，按照委员会成员分配方式的不同可分为通过链上随机数分配（如 Omniledger）、通过投票分配（如 Chainspace）及通过 PoW 分配（如 ELASTICO）。

1.4.6.1 ELASTICO

ELASTICO 是 Luu 等人在 2016 年提出的混合共识机制。ELASTICO 的核心思想是把网络中的节点分成多个小的委员会，每个委员会处理互不相交的交易集合。这些不相交的交易集被成为分片（Shard）。委员会选举是通过 PoW 进行的，完成 PoW 的节点随机分到某个委员会中，由于每个委员会中的节点数量足够小，所以委员会内部可以运行经典的拜占庭共识协议。由于委员会之间交易处理是并行的，所以 ELASTICO 几乎完成了对吞吐量的线性扩展。

为了降低委员会成员建立连接时的通信复杂度，ELASTICO 设有一个特殊的委员会，称作目录委员会。目录委员会为其它委员提供委员会分配委员会成员身份和委员会成员身份列表服务。同其它普通委员会一样，目录委员会也有 c 个成员，前 c 个找到工作量证明的节点进入目录委员会。目录委员会满员之后，后续找到工作量证明的节点会将工作量证明及身份信息发送给所有目录委员会成员，目录委员会根据提交的哈希值将其分配进相应的委员会。当所有委员会成员到达 c 之后，目录委员成员会会把每一个普通委员会的成员列表广播给相应的委员会成员。

目录委员会中拜占庭节点比例最多为 1/3，因此每个普通委员会成员将会收到最少 2c/3 个正确的成员列表，把收到的所有身份做了一个并集，就创建了一个至少拥有 c 个委员会成员的视图。

每个委员会内部运行 PBFT 算法，就本轮自身委员会内部交易集合达成共识，并将交易集合和对其确认签名发送至最终委员会，最终委员会是从所有普通委员会中随机选出的一个委员会。最终委员会会收集所有普通委员会确认的交易集合，并组成本轮的总区块，其中交易是所有之前收到的有效交易集的并集，然后内部运行 PBFT 协议，达成共识后将总区块广播到网络中。

此外，最终委员会运行随机数生成协议，生成用于下一轮寻找 PoW 的随机数集合，该过程仍需要运行 PBFT 协议，以达成对随机数集合的共识。

虽然 ELASTICO 极大的提高了区块链系统的吞吐量，但与其它的混合共识机制相比，其交易延迟过大。

1.4.6.2 Omniledger

Omniledger 是 Kokoris-Kogias 等人在 2018 年提出的混合共识机制[57]，其设计与 ELASTICO 大致相同，但是解决了 ELASTICO 存在的几个问题：（1）分片较小，在抵御 25%攻击时有较高的失败率；（2）选举不是强抗偏见的，矿工可以选择性的忽略 PoW 来得到特定的结果；（3）交易确认延迟较高；（4）不能保证跨分片交易的原子性，易锁死。

如图 12 所示，Omniledger 有一条身份链和多条交易链组成。身份链用于记录协议每个时期参与的节点和其对应的分片信息，每个时期更新一次。交易链用于记录交易信息。Omniledger 使用 RandHound 协议，将所有的验证节点分成不同组，并随机的将这些组分配到不同的交易链，验证以及共识区块。

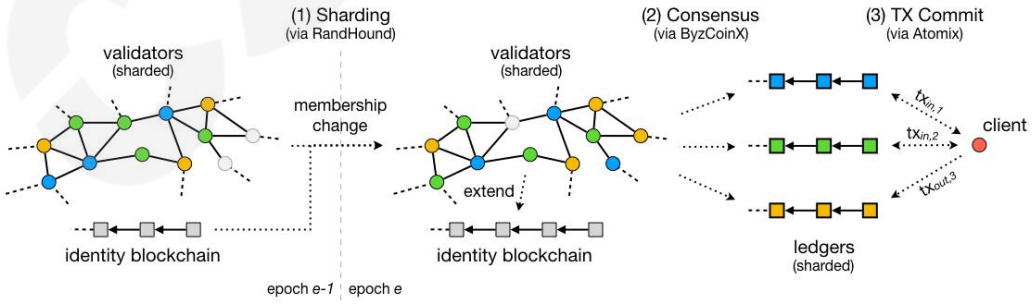


图 12 Omniledger 结构示意

Omniledger 委员会选举方式与 ByzCoin 类似，即选取大小为 n 的滑动窗口，区块处于滑动窗口内的出块者选为委员会成员。委员会通过可验证随机数（VRF）选出领导者，领

导者启动随机数生成算法 RandHound 生成随机数 r 并广播，其它委员根据随机数 r 确定自己所在的分片，组成小委员会。小委员会之间运行 ByzCoinx 协议对交易进行确认，ByzCoinx 协议相比于 Byzcoin 协议具有更好的健壮性但是牺牲了部分性能。

Hydrand[58] 改进了 Omniledger 使用的随机数生成算法改进，采用公开可验证秘密，确保了随机数的不可预测性、抗偏置性和公开可验证性，并且确保了随机数的持续生成。

1.4.6.3 Chainspace

Chainspace 是由 Al-Bassam 等在 2017 年提出的混合共识机制[59]，Chainspace 将智能合约的执行和验证步骤分离开来，并且提出了跨片交易处理的原子承诺协议（Sharded Byzantine Atomic Commit，S-BAC），S-BAC 的实质是 BFT 和原子承诺（Atomic Commit）的结合。

每个分片委员会内部运行 BFT 协议，对客户端提交事务进行决策，以暂时决定是在本地接受还是中止事务，并将其本地决策——预接受（ pre-accept ）或预中止（ pre-abort ）广播给其他相关分片。每个分片收集来自其他相关分片的响应，如果所有分片都使用预接受响应则提交事务，否则中止事务。这些分片通过向用户发送接受（ accept）或中止消息（ abort ），将此决定传递给用户以及输出分片，如图 13 所示。

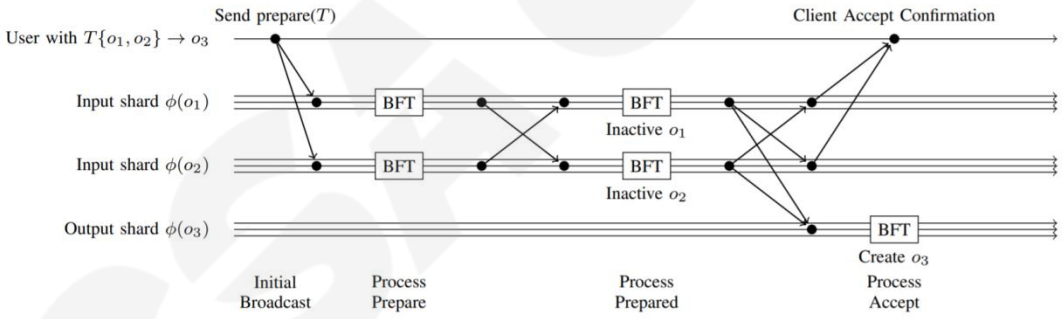


图 13 Chainspace 示意

1.4.6.4 RapidChain

RapidChain 是 Zamani 等人在 2018 年提出的混合共识协议[60]。RapidChain 基于同步网络假设，最多能容纳 1/3 拜占庭节点。Rapidchain 主要包含三个阶段，分别是启动阶段（Bootstrapping Phase）、共识阶段（Consensus Phase）和重配置阶段（Reconfiguration Phase）。

启动阶段只在 RapidChain 开始时运行一次，这个阶段是为了创建一个初始随机源，并随机选出一个特殊的委员会，叫做参考委员会（Reference Committee），再由这个参考委

员会对节点进行随机分配，构成一个个分片委员会。该阶段使 RapidChain 不需要任何可信第三方和可信启动的存在，便能够完成协议的初始化。

RapidChain 将时间划分成了一个个 epoch，每个 epoch 由共识阶段和重配置阶段组成，共识阶段又分为了多个 round，即委员会在一个 epoch 内可以进行多轮共识。在共识阶段开始时，委员会会利用当前 epoch 的随机种子随机选择出领导节点，然后委员会成员运行实用同步拜占庭容错协议，对交易进行共识。

重配置阶段会对委员会成员进行更新，网络中的其它节点可以通过寻找 PoW 加入委员会，PoW 须依赖于上一个 epoch 的随机源，参考委员会验证收到的 PoW，如果验证通过则通过有界布谷鸟规则（Bounded Cuckoo Rule）将节点随机分配到各个委员会中。有界布谷鸟规则可以使新节点尽量替换掉原本委员会中不活跃的节点。

此外，在重配置阶段，参考委员会的成员还会运行 DRG 协议（Distribute Random Generation Protocol）生成一个无偏差随机数并达成共识。

RapidChain 共识阶段采用同步网络模型，交易确认时延与网络真实时延无关，只与网络延迟上限有关，导致交易确认不能满足快速响应特性，但 RapidChain 协议其他部分采用部分同步网络模型，能够满足快速响应特性。

1.4.7 其它

除此之外，基于 Proof-of-X 的共识机制还有 Proof-of-Burn(PoB)、Proof-of-Personhood、Proof-of-authority 等共识机制，但应用都不广泛。恒星共识协议（Stellar Consensus Protocol， SCP）提出了一种联邦拜占庭协商协议（Federated Byzantine Agreement，FBA），FBA 的健壮性来源于群体切片，即由单个节点的个体信任决策共同决定系统级别的仲裁[63]。

2 共识算法安全性分析

2.1 安全建模

2.1.1 安全属性

一个安全的共识算法应满足：

（1） **一致性（Agreement/Consistency/ Integrity）**，所有的诚实节点要么都接受某个值，要么都拒绝某个值；

（2） **终止性（Termination）**，所有的诚实节点最终都会对某个值达成共识；

（3） **合法性（Validity）**，所有诚实节点达成共识的值都是由诚实节点提议的。在区块链共识算法中，更多提及的是安全性（Safety）及活性（Liveness），这两者本质上是等价的，正确性和一致性决定了安全性，而终止性就是活性。另一个评价共识算法比较重要的特性是弹性（Resiliency），即最坏情况下最多能对多少恶意节点容错、最多需要几轮共识才能终止、最大通讯复杂度是多少等。

对于一致性，又可分为强一致性（Strong Consistency）与弱一致性（Weak Consistency/Eventual Consistency），强一致性是指共识结果是确定的，具有强一致性的共识被称为确定性共识，大部分经典的共识算法都是强一致性的。弱一致性是指共识结果是不确定的，只是随着时间的推移其结果被推翻的概率会越来越小，大部分区块链共识算法都是弱一致性的。相比于弱一致性共识算法，强一致性共识算法能够保证前向安全性并实现更低的确认时延。

此外，对于区块链共识算法，还有一些重要的性质[64]：

（1） 公共前缀（Common Prefix），即对任意两个诚实节点所提交的区块链，一定能找到某个正整数 k 使得它们 0 到 k 个区块是相同的；

（2） 链质量（Chain Quality），任意诚实节点的链除去最新的 k_0 后，任意 k 个连续的区块中，恶意区块的比例不超过安全参数 u；

（3） 链增长（Chain Growth），对于任意轮 $r > r_0$，假设诚实节点 P 在第 r 轮的输出的区块链为 C_1，在第 r + s 轮输出的区块链为链 C_2，则满足$|C_2| - |C_1| \geq t \cdot s$，其中 t、$r_0$、s 为安全参数；

（4） 公平性（Fairness），诚实节点产生的区块所占主链比例与其算力（或者权益）占比一致；

（5） 前向安全性（Forward Security），达成一致的共识结果不能在之后被推翻；

（6） 快速响应特性(Responsiveness)，是指交易确认时间与网络真实时延有关，与网络时延上限无关。

（7） 自一致性（Future-Self Consistency），在任意两个时刻 r、s，任意诚实节点在 r、s 时刻，除最后 t 个区块外剩余区块不相同的概率可以忽略不计。

2.1.2 网络模型

在共识网络中有主要有两种网络结构，分别是点对点通道（Point-to-Point Channels）及点到点扩散（Peer-to-Peer Diffusion）。在点对点通道传输模型中，节点分别与其它各节

点建立可靠的通道，直接进行消息传输，在传输过程中，消息的发送方与接收方均能识别对方的身份信息，该传输模型也叫做可靠信息传输（Reliable Message Transmission）。在点到点扩散传输模型中，节点只与网络中的部分节点建立连接，消息通过广播的方法发送给与其连接的节点，收到消息的节点再以同样方式进行广播，最终将消息传递至整个网络，消息接收者无法确认消息发送者的身份，消息发送者也无法确定接收者接收消息的顺序。

对于共识协议分析及设计另一个重要的网络特征是同步性，目前对于网络主要有三种同步假设，分别是：

（1） **同步网络（Synchronous Network）**，所有消息都必然在某个可以预先确定的时间内传达，并且网络中的所有参与者都知道消息需要传递多久。在同步网络中，协议可以以轮的形式进行，在每一轮中，诚实用户发出的消息能够在下一轮之前到达其他所有诚实用户。

（2） **半同步网络（Semi-Synchrony Network）**，参与者无法确切地知道网络延迟，但消息传递时间可以由随机变量建模，该随机变量的概率分布对于协议中的参与者来说是已知的。

（3） **部分同步网络（Partially Synchronous Network）**，网络延迟存在上限，但网络中的参与者不知道该上限的确切值，时延上限不能作为协议的参数使用，部分同步网络是区块链协议分析中常用的网络模型。

（4） **异步网络（Asynchronous Network）**，仅保证诚实用户的消息能够到达彼此，但消息传递的顺序可能被打乱，消息的时延可能无限长。

2.1.3 启动模式（Setup Assumptions）

根据协议开始时所依赖的信息不同，可以有三种不同的启动模式，分别是无状态启动（No Setup）、公共状态启动（Public-State Setup）、私有状态启动（Private-State Setup）[65]。

无状态启动是指共识启动前不设置任何启动函数，没有初始状态，因此恶意节点可以在协议开始前进行预计算。公共状态启动是指协议依赖于一个特定的随机状态，该随机状态可以在协议开始时被随机取样，恶意节点可以通过某些手段干预随机状态的生成。私有状态启动是指协议启动依赖于一组状态$(s_1, ...s_n)$，n 为参与成员个数，s_i 由成员 i 秘密随机生成的。

2.1.4 安全性假设

同其他密码学协议一样，共识协议主要有两种安全性假设，分别是信息论安全性（Information-Theoretic Security）与计算安全性（Computational Security），前者假设敌手有无限的计算资源，后者假设敌手仅能对多项式复杂度的问题进行计算。

对于信息论安全性又可分为完美安全性（Perfect Security）和统计安全性（Statistical Security），前者指协议的安全性质在任何情况下都不会被破坏，后者指协议的安全性性质可能会被破坏，但被破坏的概率可以忽略不计。

2.1.5 腐化模型

腐化（Corruption） 是指敌手通过向目标节点发动攻击，获取目标节点的秘密信息，进而控制目标节点的输入输出消息，使其完全受到自身的控制。共有三种腐化模型，分别是静态敌手（Static Corruption）、T-温和敌手（T-Mildly Corruption）及适应性敌手（Adaptive Corruption）[64]。

静态敌手是指敌手只能在协议开始前选定其腐化的目标，协议运行期间，敌手不能够再去腐化其它诚实节点，其控制的节点数量也不会发生改变。T-温和敌手是指敌手需要一定的时间 t 完成对一个节点的腐化，在敌手实施腐化的 t 时间内节点仍然属于诚实节点。t-温和敌手是区块链协议分析中经常采用的腐化模型。适应性敌手是指敌手能够根据协议运行过程中搜集的信息，动态且适应性地对目标节点实施腐化，适应性敌手是三种腐化模型中最强大的。

2.2 分析方法

2.2.1 经典分布式理论

Fischer、Lynch 和 Patterson 在 1985 年证明了：在网络可靠，但允许节点失效（即便只有一个）的最小化异步模型系统中，不存在一个可以解决一致性问题的确定性共识算法，被成为 FLP 定理[1]。FLP 定理从理论层面指出为异步网络设计在任何场景下都是安全的共识算法是不可能的。

2000 年 7 月，加州大学伯克利分校的 Eric Brewer 教授在 ACM PODC 会议上提出 CAP 猜想。2 年后，麻省理工学院的 Seth Gilbert 和 Nancy Lynch 从理论上证明了 CAP 猜想。之后，CAP 理论正式成为分布式计算领域的公认定理。简单来说，CAP 定理是指一个分布式系统，最多只能同时满足一致性（Consistency）、可用性（Availability）和分区容错性（Partition Tolerance）这三项中的两项。

一致性是指分布式系统中的所有节点在同一时刻具有相同的数据。可用性是指系统中部分节点发生故障后系统仍然能响应客户端的请求。系统如果不能在时限内达成数据一致性，就意味着发生了分区，分区容错性是指系统在遇到某节点或网络分区故障的时候，仍然能够对外提供满足一致性或可用性的服务。一般来说，网络分区是一个自然的事实，分区容错无法避免，因此可以认为 CAP 的 P 总是成立，即由 CAP 定理可知，共识协议一般无法同时做到可用性与一致性。

实际上网络分区的情况极少出现，系统大多数时间能够同时满足一致性及可用性，Pritchett 基于这个观察提出了 BASE 理论[66]，其核心思想是当满足可用性时，即使无法做到强一致性，但每个节点可以采用适当的方式来使系统达到最终一致性（Eventually Consistency）。最终一致性强调的是系统中所有的数据副本，在经过一段时间的同步后，最终能够达到一个一致的状态。因此，最终一致性的本质是需要系统保证最终数据能够达到一致，而不需要实时保证系统数据的强一致性。

2.2.2 协议分析方法

目前对于计算安全性的协议主要有两种证明方法，分别是 Simulation-Based Approach 与 Game-Based Approach。Simulation-Based Approach 的主要思路是用用理想模型模拟现实模型，证明敌手无法区分模拟模型与现实模型，从而证明现实模型是安全的。Game-Based Approach 的思路是通过证明没有敌手能够实现其目标从而证明协议是安全的，其主要过程为：描述协议、描述敌手目标、描述敌手能力、构造安全模型（利用困难问题构造）、归约证实（归约到困难问题不可解上）。

Simulation-Based Approach 的主要构造形式有 Trusted-party Paradigm 、Universal Composability 及 black-box Simulatability。Trusted-party Paradigm 是由 Goldreich 等人在 1986 年提出的[67]，该方法假设存在一个可信第三方去执行协议并且可信第三方能够看到协议的所有输入。如果现实敌手执行该协议，其输出与可信第三方输出完全一致，则可认为该协议是安全的。

Universal Composability（UC）即通用可复合安全，是由 Canetti 在 2001 年提出的[68]，是解决协议组合问题的重要工具。作为一种可证安全的方法，UC 框架中定义了一整套安全性模型来证明复杂环境下组合协议的安全性。UC 安全采用模块化的思想，在 UC 框架中被证明是 UC 安全的协议，在复杂的网络环境中作为一个模块与其他协议进行组合时不破坏组合后协议的安全性，即几个分别在 UC 框架中被证明是 UC 安全的协议，组合以后仍然是安全的[69]。

目前在共识协议证明中使用比较多的是 Trusted-party Paradigm 及 universal Composability。

2.2.3 区块链建模

Garay 等人在 2015 提出了 Bitcoin Backbone Protocol[44]，最早提出了半同步假设下的线性表结构区块链的三个基本共识特征：公共前缀（Common Prefix）、链质量（Chain Quality）和链增长（Chain Growth）。Bitcoin Backbone Protocol 从理论上解决了在只有随机出块节点竞选而没有显式的 BFT 委员会投票机制情况下，比特币“最长链法则（The Longest-Chain-Rule）”的有效性。同时作者证明了任何满足这些属性的区块链协议都可以被用来构造公共账本，公共账本满足：（1）持续性（Persistence），如果消息被添加到公共账本，它永远不会被删除；（2）活性（Liveness），如果所有诚实参与者想要向账本添加一些消息，那么最终消息应该出现在账本上面。目前已知的链式区块链共识协议分析大多基于 Bitcoin Backbone Protocol 提出的这三个基础特性。

2017 年 Garay 对 Bitcoin Backbone Protocol 协议进行了扩展[70]，对比特币的难度调整进行了理论分析，并对公共前缀（Common Prefix）和链质量（Chain Quality）的定义进行了适当的修改，证明了可以基于满足上述两个共识特征的区块链构建安全的公共账本。此外，Kiayias 和 Panagiotakos 证明了只要满足 Chain Quality 与 chain Growth 就可以不依赖具体协议地证明协议的活性（Liveness）[71]。

在 2017 年，Rafael Pass 等人对异步网络环境下的区块链协议进行了系统性的分析以及抽象，给出了形式化的模型和主要定理[72]。根据这些模型和定理，可以准确地理解区块链协议各种安全属性的依赖条件以及边界。这篇论文为后续区块链协议的安全性证明打下了基础，包括 Ouroboros 和 DFINITY 在内的热门项目的论文均以该论文中提出的模型和部分结论为基础进行安全性证明。

2.3 攻击方法

本节对共识算法安全威胁进行全面的调研，并给出分类和进一步的分析。目前已知有多种针对不同共识算法的安全威胁，其中有理论上的安全风险，也存在实际部署中容易被利用的漏洞，下面初步例举如下：

2.3.1 女巫攻击（Sybil Attack)

女巫攻击是在对等网络中节点通过创建多个虚假身份标识进而提高系统中所控制节点的比例，从而对系统进行破坏。通过女巫攻击可以提高恶意节点在共识系统中的投票权，

使其超过系统所能容忍的最大恶意节点数量，从而破坏共识。如果将传统的 BFT 类共识协议直接应用于公开网络环境，由于失去了准入机制的保护，协议将会因为女巫攻击的存在而不可用。为了解决这个问题，就需要引入基于 PoW/PoS/PoSpace 等的准入机制，提高节点复制身份的成本，从而避免女巫攻击的发送。

2.3.2 自私挖矿（Selfish Mining）

自私挖矿是 Eyal 等人在 2014 年针对 PoW 共识协议提出的一种攻击手段[73]。传统观点认为，比特币的挖矿协议是激励相容的，它可以抵御少数群体的合谋攻击，并激励矿工按照协议规定的方式进行挖矿。Eyal 等否定了这个观点，并给出了一种挖矿策略（自私挖矿），可以使恶意矿工获得比诚实挖矿更高的收益。自私挖矿会造成区块链网络分叉增加，影响网络的安全性。

自私挖矿的基本策略是恶意矿工故意延迟公布其计算得到的新块，并创造一条私有分支，基于私有分支继续挖矿，而诚实矿工则继续基于公开的分支进行挖矿，当私有分支长于公开分支时，恶意节点将继续隐藏私有分支，但当公共分支接近私有分支的长度时，恶意节点将公布私有分支，从而使公开分支上的最近几个块无效。

以上过程可以用一个马尔科夫链描述，通过自私挖矿期望收益如图 14 所示，可见当算力超过 1/3 时，恶意节点可以通过自私挖矿来提高收益。

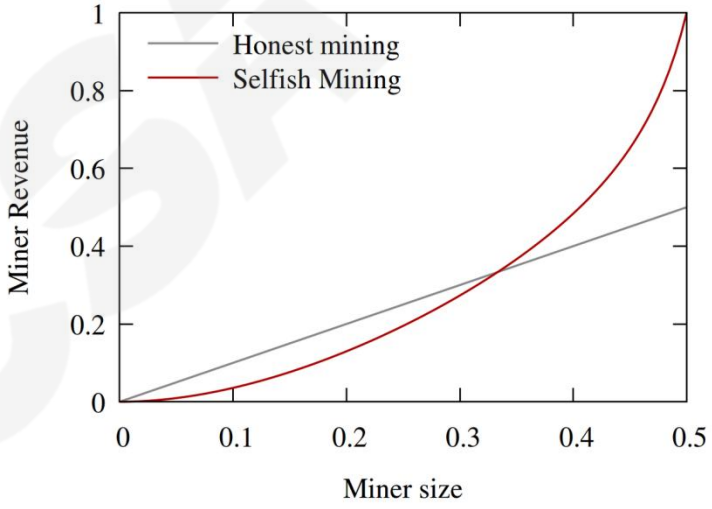


图 14 自私挖矿收益示意

Nayak 等在 2016 年提出了一种新的挖矿策略“Stubborn”，该策略对“自私挖矿”策略进行了扩展，相较于使用“自私挖矿”策略，使用该策略的恶意节点的收益将提高 13.94%。作者还进一步对“Stubborn”策略进行了优化，并提出了两个新的策略，即“the

EqualFork Stubborn”和“Trail Stubborn”，这两个策略进一步提高了恶意节点的挖矿收益。Carlsten 研究了交易手续费对于“自私挖矿”策略的影响[75]。Göbel 等人进一步扩展了“自私挖矿”模型，将恶意节点与诚实节点之间的通信时延加入了模型[76]。基于此模型，作者提出了一种通过监测“孤立块”比例检测“自私挖矿”异常行为的方法。Houy 则考虑了一个更加通用的假设，利用贝叶斯博弈公式对“自私挖矿”矿工在策略中的选择行为建模，进一步优化了挖矿策略[77]。

PoW 共识机制及以 PoW 为基础的混合共识机制都会受到自私挖矿的影响，一些针对自私挖坑的解决方案被提出。Solat 在 2016 年提出了 Zeroblock[78]，要求一个块在块的时间戳之后的特定时间间隔内被其它节点接受，否则该块就会过期无效，这种机制在一定程度上可以防止恶意节点进行自私挖矿。Pass 等人提出了 Fruitchain[32]，通过将出块奖励平坦到一个个小的水果区块解决了自私挖矿问题。Zhang 等通过设计基于分叉率的难度调整算法，使节点无法通过自私挖矿提高收益[35]。

2.3.3 长程攻击（Long range attack）

长程攻击（Long Range Attack）则是权益证明协议中最大的威胁之一。由于权益证明协议有弱主观性并且能进行无代价模拟，这种攻击比在工作量证明协议中更为危险。长程攻击就是攻击者创建了一条从创世区块开始的长区块链分支，并试图替换掉当前的合法主链，该分支上可能存有和主链不同的交易和区块，所以这种攻击又被称 替换历史攻击或历史覆写攻击。

弱主观性指的是区块链网络中的新节点或是长期离线的节点加入到网络中时无法分辨出哪些分支是从属于主链的。基于 PoW 的共识协议可以通过最长链原则来选取主链，但是当遭遇 51%攻击时这个原则也会失效。但 51%攻击的代价非常高昂，因此弱主观性并不能对基于 PoW 协议造成实质威胁。但是对于基于 PoS 的共识协议，从头构建一条链几乎是零代价的，仅仅依靠最长链原则不足以判断哪一条是主链，从而就造成了攻击的可能。

目前总共有三种不同类型的长程攻击，分别是简单攻击（Simple Attack）、变节攻击（Posterior Corruption）和权益流损（Stake Bleeding）。

简单攻击是指通过伪造区块时间戳达到伪造的链具有更大链长度的目的，该攻击适用于不验证时间戳的权益证明协议区块链中。

变节攻击是指对于已经退出区块链的节点，其私钥仍然可以用来产生历史区块，攻击者可以收买这些私钥伪造区块，增大其覆盖主链的概率。变节攻击可以使用密钥演进加密技术（Key-Evolving Cryptography）和移动检查点技术（Moving Checkpoint）应对。

权益流损攻击是指为了增加攻击的成功率，攻击者可以拖延主链的正常运行，如果攻击者持有的权益占比足够多，这种行为可能会演变为一次活性冻结攻击（Liveness Denial Attack）。每当攻击者选为主链上的出块节点时，都会跳过出块，但这并不意味着别的节点可以替代其工作。相反，在该区块位置处不会有新的区块加入到主链中。而攻击者在自己的分支上积极出块。通过采用这样的策略，一方面拖延主链的出块速度，一方面在分支链上尽可能多的出块，攻击者最终可以在自己的分支链上获得绝大部分的权益，并且比主链更快地产生区块，最终覆盖主链。权益流损攻击一般需要的时间比较长。权益流损攻击可以通过采用移动检查点的策略应对。

2.3.4 Difficulty Raising Attack

Difficulty Raising Attack 是由 Bahack 在 2013 年提出的一种攻击方法[79]，该方法允许任意算力的攻击者只要等待足够长的时间就可以以 100%的概率覆盖主链。该攻击方法的主要思路是攻击者私下构造一个区块，该区块难度是主链分支上所有区块难度的和。只要时间足够长，攻击者一定能找到一个这样的区块从而覆盖掉主链分支。

由于攻击者需要根据主链的出块情况不断地调整目标难度，控制难度调整速度上限可以在一定程度上缓解该攻击，实际是目前大部分带难度调整的区块链共识协议都规定了一次调整所能调整的上限。此外由于该攻击所需要的时间非常长，在实际场景中并不太可能发生。

2.3.5 无利益攻击（Nothing at Stake）

除长程攻击外，无利益攻击是基于 PoS 共识协议所面临的另一个主要威胁。无利益攻击问题的本质是“作恶无成本，好处无限多”。因为在基于 PoS 共识的协议中出块是无成本的，当在系统出现分叉的情况时，出块节点可以在“不受任何损失”的前提下，同时为多条链出块，从而有可能获得“所有收益”，极大破坏共识的收敛性，甚至导致共识无法收敛。该攻击对基于 PoSpace 的共识协议也有效。

解决无利益攻击的一个主要手段是引入惩罚机制，对同时在多个分支上出块的节点进行经济惩罚（Slashing），从而迫使节点选择一个分支出块，加快收敛。

2.3.6 权利压迫攻击（Grinding Attack）

权利压迫攻击是指恶意节点通过影响出块节点的选举过程，加大自己被选为出块节点的概率。在基于 PoS、PoSpace 的共识协议中，随机性来自于链本身的原始数据。恶意节点在出块时，可以通过尝试生成不同的区块以找到对自己有利的随机数，增大自己后续选为出块节点的可能性。

可以通过减少采样频率的方法降低权利压迫攻击的影响，比如每隔 10 个区块采样一次，10 个随机数由采用值不断 Hash 生成，这样恶意节点仅能影响其中一个随机数的生成。

2.3.7 双花攻击（Double Spending Attacks）

双花攻击是指攻击者通过某些方法将之前确认的交易推翻，双花攻击一般有三种方法：（1）Race Attack，对于同一笔钱攻击者发送两笔转账交易到区块链网络，其中一笔转向自己，由于转向自己的交易附有较高的手续，有更高概率被矿工打包；（2）Finney Attack，恶意节点通过预先挖到的区块推翻发出的转账交易；（3）51% Attack，当恶意节点的算力超过 51%时，可以通过创建更长链推翻之前区块上的交易。

尽管不同的共识算法试图缓解此漏洞并采用不同的机制来解决该漏洞，但在区块链系统中无法完全避免重复支出，并且理论上可能一直在发生。

2.3.7 交易拒绝攻击（Transaction denial attacks）

交易拒绝攻击（Transaction Denial Attacks）是指攻击者阻止某笔交易成功完成，破坏了区块链系统的活性。交易拒绝攻击可以通过控制 P2P 网络，进而使交易无法被正常广播，也可以通过通过控制矿工节点，使交易无法被打包。

2.3.8 无法同步攻击（Desynchronization Attacks）

无法同步攻击是指攻击者通过某些手段使系统中的节点无法与其它节点同步。该攻击通常是发生在基于 PoW、PoS、PoSpace 共识的开放网络中，由于 BFT 类共识协议其节点不会直接暴露在开放网络中，且节点之间一般直接连接，因此共识节点很难产生无法同步的问题。

2.3.9 日蚀攻击（Eclipse Attack）

区块链网络的正常运行依赖于区块链节点间路由信息的共享。Eclipse 攻击是指攻击者通过侵占节点的路由表，将足够多的虚假节点添加到某些节点的邻居节点集合中，从而将这些节点“隔离”于正常区块链网络之外。当节点受到 Eclipse 攻击时，节点的大部分对外联系都会被恶意节点所控制，由此恶意节点得以进一步实施路由欺骗、存储污染、拒绝服务以及 ID 劫持等攻击行为。

Eclipse 攻击者通过不断地向区块链节点发送路由表更新消息影响区块链节点的路由表，试图使普通节点的路由表充满虚假节点。当区块链节点的路由表中虚假节点占据了较高的比例时，区块链网络的正常行为，包括路由查找或者资源搜索，都将被恶意节点隔绝。

Eclipse 攻击破坏了网络的拓扑结构，减少了节点数目，使得区块链网络资源共享的效率大大降低。在极端情况下，Eclipse 攻击能完全控制整个区块链网络，把它分隔成若干个区块链网络区域。

2.3.10 DDOS 攻击

DDoS 攻击是一种对区块链网络安全威胁最大的攻击技术之一，它指借助于 C/S 技术，将多个计算机联合起来作为攻击平台，对一个或多个目标发动攻击，从而成倍地提高拒绝服务攻击的威力。

根据攻击方式的不同，基于区块链的 DDoS 攻击可分为主动攻击和被动攻击两种。基于区块链的主动 DDoS 攻击是通过主动地向网络节点发送大量的虚假信息，使得针对这些信息的后续访问都指向受害者以达到攻击效果，具有可控性较强、放大倍数高等特点。这种攻击利用反射节点在短时间内发送大量通知信息，不易于分析和记录，并且可以通过假冒源地址避过 IP 检查，使得追踪定位攻击源更加困难。此外，主动攻击在区块链网络中引入额外流量及虚假的索引信息，会降低区块链网络的查找和路由性能、影响文件下载速度。

基于区块链的被动 DDoS 攻击通过修改区块链客户端或者服务器软件，被动地等待来自其它节点的查询请求，再通过返回虚假响应达到攻击效果。通常情况下，需采取一些放大措施增强攻击效果，如：部署多个攻击节点、在一个响应消息中多次包含目标主机、结合其它协议或者漏洞等。被动攻击属于非侵扰式，对区块链网络流量影响不大，通常只能利用局部的区块链节点。

如果共识协议中出块节点的选择可以被预测，则可以针对出块节点进行 DDoS 攻击，从而达到拒绝打包某些交易或者禁止某些节点出块的目的，破坏区块的安全性及活性。

2.3.11 51%攻击

51%攻击即网络中恶意节点的算力或权益占到了 50%以上。对于 PoW 机制，恶意节点掌握了全网络 51%的算力，那么该攻击者构造一个合法区块的平均时间会少于其他矿工，从而在相同的时间内可以构造出更多区块，并以最长合法链被网络接受，这样整个网络都可能在该恶意节点的控制下。在 PoS 算法中，51%攻击转变为掌握网络中的大多数权益。在拜占庭类算法中，控制 33%的节点即可阻止达成共识，控制 2/3 的节点就可操纵共识结果。

2.3.12 贿赂攻击

贿赂攻击是指在区块链中存在一个拥有足够资源的贿赂者，通过额外的奖励激励其他节点采取特定行动的攻击行为，也就是“收买”现有节点对区块链进行攻击。该攻击可以不用花大成本去掌握 51%的算力，就可以对区块链造成分叉。可以引入保证金和惩罚措施防范此类攻击。

2.3.13 历史多数攻击（Past Majority Attacks）

历史多数攻击是指历史上对某区块链拥有控制权（出块权）的组织或个人，利用自己历史上存在的对该网络的控制权，从这个历史时间点开启新的分叉，甚至覆盖主链。

该攻击对基于 PoS 的共识协议影响较大，因为在历史上存在权益的节点，从历史时间点构造一条新的分叉几乎不费任何代价。

2.3.14 BGP Hijacking

BGP 劫持攻击即利用 BGP 操纵路由路径，如误导或拦截流量等，进而可以破坏区块链中共识机制、交易等信息。目前有数据统计大多数比特币节点都保管在少数特定的几个互联网服务提供商中，比特币网络的集中化更容易遭受 BGP 劫持攻击。

2.3.15 P+Epsilon 攻击

P+Epsilon 攻击也是贿赂攻击。假设每个矿工都需要投票给 0，若达成共识则每个矿工都可以获得 P 收益，若有人投票给 1 则收益为 0。攻击者进入系统，告诉每个矿工若他投票给 1 而其他人不投票，则他可以额外获得 Epsilon 收益。每个矿工由于并不知道攻击者也告诉了其他矿工，因此有很大概率会选择投票给 1 增加收益，最终所有矿工都投票给 1，达成了不利于区块链的共识。而此时攻击者甚至都不需要支付贿赂费用，对攻击者造成了双赢的局面，但对区块链影响重大。

2.3.16 冷启动攻击

PoS 机制中，由于持币量会对挖矿难度产生影响，因此当一个基于 PoS 的代币系统启动时，早期获得代币的持有者就会没有动力去花费或者转移代币。而且这些持有者会更容易挖矿，因此会出现初始代币流通性不好的问题。

2.3.17 币龄累积攻击

币龄攻击针对 PoS 和 DPoS 算法，节点获得记账权的可能性与账户中持币的数量和每个币的持币时间有关，当节点拥有的币越多，持币时间越长时，获得记账权的可能性就越

大。这样就会导致部分节点买入一定数量币后，持有足够长时间后达到51%以上的算力，从而控制整个网络。

2.3.18 预计算攻击

预计算是指当PoS中的某一节点有了一定量算力之后，就有能力控制挖矿参数使得挖矿难度在自己算力范围内。

2.3.19 藏块攻击（Block withholding attack）

藏块攻击是指攻击节点将一部分算力加入矿池挖矿，另一部分算力独自挖矿，加入矿池的那部分算力正常响应矿池发来的挑战，但是如果找到可以出块的答案，则不发送给矿池，由于找到可以出块的答案的概率比较小，不会影响矿池计算其在矿池中所占的份额，这样做会导致整个矿池的收益降低，但是会降低挖矿难度，提高攻击者独立挖矿算力的收益，从而使其整体收益提高，破坏了区块链共识协议的公平性。

3 共识算法安全测试方法和标准

本节从共识算法理论和实现两方面展开安全测试和分析，理论分析主要从算法本身展开分析，不涉及算法的代码实现和具体应用部署环境。实现分析从算法的具体参数、代码实现、应用部署等都方面进行安全分析和测试。

3.1 共识算法安全理论分析和模拟

本节主要考虑算法本身和参数方面是否存在安全缺陷，通过模拟方式测试、验证共识算法在网络部署时的安全性。

3.1.1 共识算法安全理论分析

对于共识算法理论上是否满足安全性，主要考察共识算法是否满足一些属性，Bonneau对文献[44][81][82]的工作进行了总结，提出了共识算法一般需要满足以下五个安全特性[80]：

（1） 最终一致性，即在任何时候，所有合法节点都对最终能成为有效区块链的前缀链达成共识。但是不能要求任何时候最长的链最终都能成为有效区块链的前缀，因为存在临时分叉，有些块会被丢弃。

（2） 指数收敛，即长度为n的分叉出现的概率为$O(2^{-n})$，这会给用户高度信心，区块被·确认的规则将使用户的交易永远包含在有效区块中。

（3） 活性，即新的区块会持续添加到链上，并且拥有合理交易费的有效交易会包含在区块中。

（4） 正确性，即最长合法链只包含有效交易。

（5） 公平性，即拥有占总算力α的矿工将挖出约α比例的区块。

有了以上标准，就按照此标准来从理论上初步判定一个共识算法是否具备安全性。共识算法中包含许多参数，这些参数一般都会影响共识算法的安全性，Bonneau 在文章中列举几个重要参数：

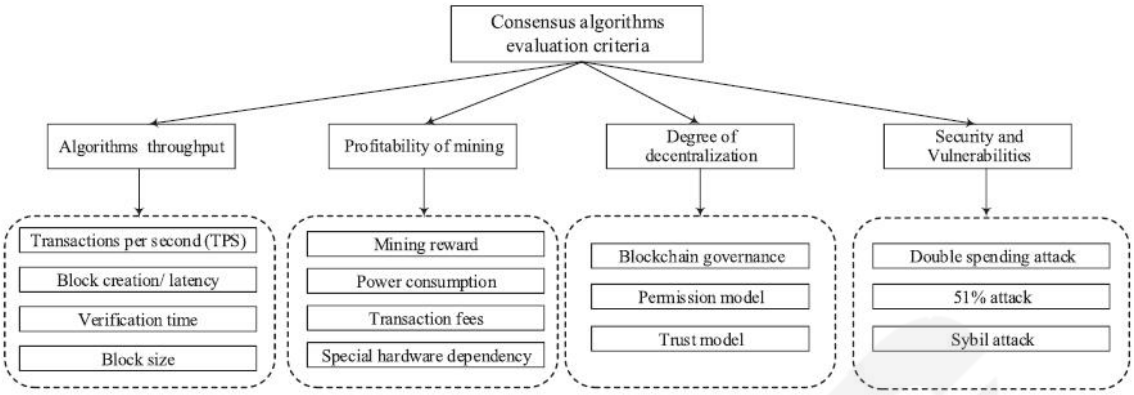
（1）出块间隔和难度调整窗口：出块间隔定义了矿工将区块写入区块链中需要等待的时间，出块间隔越短，确认交易就会越快，区块过期的可能性就越高，因此可以观察其出块时间判定共识算法的安全性。共识算法会自动调整对抗性难题的难度，以便在固定时间内出块，难度较低就会使网络中的块数量较多，出块速度较快，难度较高就会导致块数量较少。若出块速率过快，那么受到网络延迟的影响，矿工通常会在这些块被传播之前就找到冗余块，较容易出现分叉。若出块速率过慢，那么就增加了用户等待事物确认的时间。因此，难度调整窗口会影响攻击者对最长合法链的攻击能力，并且为了防止双花攻击，使得商家可以安全接受交易，还需要调整交易确认需要等待的块数。

（2）块大小限制：通常情况下，区块有大小限制，比如比特币的块大小为 1MB。随着交易量稳步增长，这一限制可能很快就会达到。一旦达到该限制，交易者就需要支付更高的交易费用竞争资源。攻击者可能会利用块大小限制发起 DDoS 攻击和女巫攻击，通过发起数笔交易来使合法用户受到阻塞。通过观察块大小限制可以初步判定共识算法安全性。

（3）货币政策：共识算法中通常会规定货币政策，比如比特币会规定新货币的铸造速度，还有一些 PoS 算法会规定币龄上限。货币政策通常会影响初始代币的价格，若是通货紧缩的货币政策，那么最终可能会导致没有人愿意购买新的代币，囤积该币会更加有利可图，造成流通性不好的局面。而一些 PoS 算法中，币龄与算力成正相关，最终可能会导致攻击者持有 51%的算力发起攻击。

3.1.2 安全性与恶意攻击类型

攻击行为一般发生在交易或者区块链前后的共识阶段。在共识过程中，为了保证区块链的稳定性，共识安全性的评估是非常有必要的。倘若区块链的安全性出现问题，那么即便区块链吞吐量（TPS）再高或者共识节点再多，区块链也无法保证其交易数据不可篡改和真实有效性。本小节将从理论的角度分析共识算法安全性和恶意攻击之间的关系。



对共识的恶意攻击主要类型在 2.3 节已经介绍.

3.1.3 应对策略

对于共识的攻击，主要有以下几种应对策略：

1）增加创建节点的成本。

提高创建身份的成本可以有效减轻恶意攻击的风险。例如，在燃烧证明算法中，用户需要购买然后燃烧一定量的代币（通过将一些硬币发送到不可逆的地址）验证其身份。在堆栈证明中，用户需要在堆栈中放入一些代币。在工作量证明中，用户需要拥有并花费一些计算能力。该策略的主要挑战是如何找到理想的身份创建成本，以有效减轻恶意攻击的风险，并且不限制普通人加入网络。创建节点的成本也可能随时间变化，并且区块链共识应能根据网络需求修改其所使用的成本。

2）引入某种类型的信任。

抵御恶意攻击的第二种常见方法是在为节点加入区块链引入额外的信任机制。这种信任机制可以通过简单的两步验证/电子邮件/ SMS 或要求一组管理员进行验证的方式实现。基于用户的 IP 地址进行一些限制以及防御僵尸网络的其他常见方法也可用于提供这种信任。

3）赋予身份不平等的权力。

为用户和节点赋予不同的权限是防御恶意攻击的另一种方式。例如，在权重证明算法中，根据多个参数（如用户帐户的使用期限、与之进行交易的唯一身份用户的数量、用户拥有的代币数量）为每个帐户赋予权重以及交易数量。给定的权重赋予每个用户相关的投票或参与权。但是，这种不平等的权力分配使该系统成为精英制而不是纯粹的民主制，对于新用户而言可能并不友好。不同的共识机制使用不同的方法或它们的不同组合来防御恶意攻击。没有“最佳”策略，每个策略都有自己的优缺点。

4）根据业务场景切换共识算法

评价共识算法的一般有四个维度：安全性、扩展性、性能效率和资源损耗。不同共识算法所侧重的方向也各不相同。因此，当面对不同的场景时，为了保证安全性，可以在全网同时采用多种共识算法，通过多级共识确认交易，以及根据业务场景选择合适的共识算法。

3.1.4 共识算法安全模拟

通过 3.1.1 节对共识参数作理论分析，可以初步判定共识算法是否存在安全漏洞，为了更好的对共识算法进行分析，还可以用模拟的方式测试共识算法的安全性。

3.1.4.1 经典共识算法安全模拟

一个直观的想法是运行实际的区块链应用程序分析区块链的性能，比如 Castro 使用真实系统将 PBFT 共识算法与其他算法进行了比较[84]。另一种思路是通过建立数学模型对 PBFT 算法进行模拟，以分析其更多的性质。

Kai Zheng 等人提出使用连续时间马尔可夫链（CTMC）模型模拟 PBFT 的共识过程[85]，该 CTMC 模型包括 5 个部分：请求过程、预准备过程、准备过程、提交过程和答复过程。请求过程是客户端将请求发送给主节点；预准备过程是区块链中一个主节点被选中，其他节点变为副节点，主节点要向副节点广播消息；准备过程是副节点计算哈希摘要并且向所有节点广播；提交过程是如果主节点和副节点都收到了 2f 的哈希摘要，并且摘要和自己的相同，就进行提交；答复过程是如果一个节点收到 2f+1 的提交信息，就可提交新区块。CTMC 模拟了此过程，整个模型中有 1 个客户端，1 个主节点和 6 个副节点。作者主要考察了网络传播速度、客户端发送或接收消息的平均时间、主节点向其他节点发送消息的时间、副节点向其他节点发送消息的时间对于能否达成共识一致的影响。实验结果表明，这四者均是速度越快、发送或接收消息的平均时间越短，达成共识一致的可能性就越高。其中，网络传播速度对于达成一致的影响最大，其次是客户端、主节点，副节点对达成一致的影响最小。

Sukhwani 等人使用随机奖励网络（SRN）对 PBFT 共识协议进行了建模[86]，主要对节点之间共识消息的传输时间、处理传入共识消息的时间和准备下一阶段共识消息的时间进行评估。作者假设：

（1） 在块事务开始之前每个领导节点就已经被选择好，并且在三阶段协议执行时不会改变；

（2） 每个验证节点处理消息的速度相同；

（3） 所有节点之间的消息传输率相同；

（4） 验证节点在执行三阶段协议时不会失效。

作者之后对影响 PBFT 共识算法的因素进行了分析。首先是敏感度分析，作者发现与为下一阶段准备共识消息相比，处理传入共识消息和提交消息的速度若较慢的话，对各节点达成共识会有较大影响。之后作者又考虑了节点数量对达成共识时间的影响。当节点数量增多时，基本上达成共识的时间也需要增加，这是因为节点增多会导致准备和提交阶段中消息的排队延迟增加。但是如果传输延迟比处理消息或排队的时间大一个数量级或者两倍，则随着节点数量的增多，达成共识的时间不会明显增加。因为 PBFT 协议是基于消息传递的算法，因此共识消息传递的延迟对于能否达成共识较为重要。

Halalai 等人提出了一个基于模型检查和仿真技术的结合模型 HyPerf[87]，用于评估 BFT 协议的性能。模型主要包括客户端和副本，模拟了它们的状态以及客户端和副本之间消息通信过程。同时作者定义了 BFT 共识协议的安全标准：

（1） 请求/响应对应，对于所有的客户端，接收到的响应都对应于之前发出的请求。

（2） 状态可达性，对于所有正确的副本，可以通过一系列客户端请求到达每个状态。

（3） 线性化，对于所有正确的副本，所有的请求都按照客户端看到的顺序执行。

（4） 状态一致，所有正确的客户端最终都会受到响应，所有的副本最终都能达到一致状态。

但是上述模型只考虑了各节点最终的状态，并没有考虑时间消耗，因此作者为客户端和每个副节点的状态中都增加了时间消耗。增加时间消耗后，该模型就可以考虑对请求的响应时间，并把此增加到上文所述的安全性能中：

（5）时效性。对于所有的客户端，每个请求都需要在一定时限内完成。

之后作者用该模型对 PBFT 算法进行了测试，主要测试了协议的时耗问题，并且考虑了有恶意行为的情况，即有一个副节点是拜占庭节点。结果表明这样的情况对于 PBFT 没有很大影响，系统还是与正常情况一样达成了共识。

3.1.4.2 区块链共识算法安全模拟

由于基于区块链的共识协议多应用于开放网络，环境复杂，难以通过实际运行协议来对协议进行安全分析，因此多通过建立数学模型进行模拟。Gervais 使用马尔科夫决策过程（MDP）对挖矿行为进行了模拟，将其建模为一系列步骤[83]。MDP 是离散时间随机控制过程，针对一些决策的输出结果部分随机而又部分可控的情况，给决策者提供一个决策制定的数学建模框架。与其他文献中的做法相似，要将区块链系统建模为 MDP，就要先

将区块链状态进行编码，并将参与者的可用决策编码为多个动作，并且用状态转移矩阵描述每个（状态，动作）的概率分布。MDP 中描述的区块链状态包括可能影响诚实矿工和攻击者的所有信息，如竞争链的长度、攻击者挖出的未发布的区块数量等。诚实矿工会根据自己的算力，以概率分布开采新的区块，而攻击者会决定是否要发布新区块、发布多少区块以及要对哪条链进行挖掘，而这些事件都会触发 MDP 状态转换。通过对该过程进行模拟，对攻击者的攻击过程进行建模，可以及时发现共识协议的弱点。

在 MDP 过程中观察以下参数：

（1） 孤块产生速率：孤块产生速率 r_s 会考虑不同的块大小、出块间隔、网络延迟、消息传播机制和网络配置（如节点数）。

（2） 挖矿算力：α指攻击者算力占系统总算力的那部分，剩余部分的算力掌握在诚实节点中。

（3） 挖矿成本：对抗挖矿成本$c_m \in [0, \alpha]$对应于攻击者的预期挖矿成本（即总挖矿成本，包括硬件成本、电力、人力等），并以块奖励的形式表示。例如，如果 $c_m = \alpha$，则攻击者的挖矿成本等于其挖矿算力乘区块奖励，即挖矿成本完全由诚实矿工挖矿的区块奖励所覆盖。

（4） 区块确认数：区块确认数 k 对应需要确认交易以使商家接受交易的区块数。

（5） 传播能力：传播参数 γ 表示攻击者在网络中的流通性，即当攻击者和诚实节点在网络中同时释放区块时，网络接收攻击者区块的概率。

（6） 日蚀攻击的影响：该模型考虑了日蚀攻击。在这里，我们假设诚实矿工会收到孤块产生速率的影响，但是攻击者及其同谋却不会在孤块上继续挖矿。这是因为攻击者可以对任意的区块进行攻击，并且当攻击者选择了一条诚实的链后，有很小的几率去开采孤块。因此在实际中，攻击者比诚实矿工挖到孤块的概率小得多。诚实的网络因为受到网络传播和延迟的影响，因此会有更高的孤块产生速率。

为了定义最佳的双花攻击策略，作者定义了与最小交易值相对应的双花攻击金额v_d，该双花攻击金额是使得双花攻击的收益大于诚实挖矿获得的收益的最小交易值。v_d可以作为量化双花攻击下共识算法安全性的可靠指标，即如果诚实矿工挖矿获得的奖励大于不诚实行为的奖励，那么商家可以安全地接受价值v_d的付款交易，因为这样的价值被认为是安全的。但是如果攻击者行为在经济上有更多回报，那么商家会意识到相关的双花攻击风险和矿工的相关激励措施。

现在使用 MDP 模型$M :=< S, A, P, R >$模拟区块链系统，其中所有其他参与者都遵循共识协议，S 表示空间状态，A 表示动作空间，P 表示随机转移矩阵，R 表示奖励矩阵，M 表示 MDP 过程。在模型中，攻击者可以采取以下动作：

（1） 接受：攻击者接受诚实网络的链，此动作其实是攻击者重启发动攻击的开始，若攻击者发现赢取诚实链的可能性较小，那么此操作是适当的。

（2） 覆盖：攻击者比诚实节点发布的块多了一个块，因此可以覆盖有冲突的区块。当攻击者的秘密链比当前已知的公开链长（即$l_a > l_h$），并且攻击者发布$l_h + 1$ 个区块以用自己的秘密链替换公开的诚实链时，就会出现覆盖的情况。如果攻击者利用诚实节点的挖矿能力，则攻击者可能会使用诚实节点的b_e个区块实现覆盖操作。

（3） 竞争：攻击者发布与诚实节点发布的一样的区块，触发两条链之间的竞争，而不是发布比诚实节点多的区块进行覆盖。

（4） 等待：攻击者继续在自己的秘密链上进行挖矿直到挖到一个区块为止。

（5） 退出：此操作仅在研究双花攻击时才有意义，因为它对应于具有k个确认的成功双花攻击，并且仅在$l_a > l_h$并且$l_a > k$时才可行。

状态空间S定义为形式为$(l_a, l_h, b_e, fork)$，其中l_a表示对手链的长度，l_h表示诚实链的长度，b_e表示通过日蚀攻击挖到的区块，$fork$可以用三个值表示，分别是不相关、相关和活跃：

（1） 相关：相关表示诚实节点找到了最后一个区块，以及如果$l_a > l_h$则竞争操作可用，$(l_a, l_h - 1, b_e, \cdot)$的状态会变为$(l_a, l_h, b_e, related)$。

（2） 不相关：当攻击者找到最后一个块时，前一个块可能已经到达网络中的大多数节点，因此攻击者无法进行竞争操作，$(l_a - 1, l_h, b_e, \cdot)$的状态会变为$(l_a, l_h, b_e, irrelated)$。

（3） 活跃：当攻击者确定执行竞争操作时，该状态被标记为活跃，即当前网络正在分裂并且正在确定最长合法链。

在该模型中，每一次状态的转变都对应着一个区块的创建，因此每次状态转变都意味着对诚实节点、攻击者或者日蚀攻击受害者的一次区块奖励。

给定攻击者的挖矿算力α，初始状态$(0,0,0, irrelated)$会以概率α转换为$(1,0,0, irrelated)$，即攻击者找到了一个区块。如果诚实节点发现了一个非孤块，则状态

会变为$(0,1,0,related)$。如果诚实节点发现了一个孤块，则状态会保持$(0,0,0,irrelated)$，因为孤块不计入最长合法链中。

此外 R. Zhang 等人对 MDP 进行了改进[35]，将 MDP 模型扩展到了拜占庭式攻击者，也就是不仅考虑经济利益方面的攻击，同时还考虑审查攻击或主链质量攻击，并且还引入了人工智能技术分析协议的安全性，给定攻击者的攻击目标，可以系统地找到共识协议中该方面的安全漏洞，从而有利于迭代地改进协议。

3.2 共识算法安全渗透测试和代码安全审查

3.2.1 安全指标

除了对共识算法进行理论分析和模拟外，也需要测试算法实现中能否保证正确性和安全性。R. Zhang 等人给出了四个指标，用以衡量在实现过程中算法的安全性[35]：

（1）链质量：主要用于衡量替换诚实主链的难度，用区块链中诚实矿工挖出的区块数量除以诚实矿工和攻击者共同挖出的区块数量计算得出。假设攻击者控制了总挖矿能力的α，则链质量Q定义为诚实矿工拥有的那部分挖矿算力 的下界。将B_c和B_a分别定义为诚实矿工和攻击者挖出的区块总数，s是攻击者的攻击策略，那么我们有：

$$Q(\alpha) = \min_{s} \lim_{t\to\infty} \frac{B_c}{B_a+B_c},$$

理想情况下，$Q(\alpha) = 1-\alpha$，即攻击者获得的区块数最多与他们的挖矿能力成正比。$Q(\alpha)$越大，说明诚实矿工挖出的区块数量更多，该共识算法被攻击者替换主链的难度就越大。

（2）激励兼容性：主要用于衡量共识算法对自私挖矿的抵抗力，用区块链中诚实矿工获得的奖励除以诚实矿工和攻击者总共获得的奖励计算得出，计算方式如下：

$$I(\alpha) = \min_{s} \lim_{t\to\infty} \frac{\sum R_c}{\sum R_c+\sum R_\alpha},$$

其中$\sum R_c$和$\sum R_\alpha$分别是诚实矿工和攻击者获得累积收益，激励兼容性与链质量有相同的理想值 $1-\alpha$。若该值越大，说明诚实矿工获得的奖励更多，矿工更愿意按照规定的共识协议进行挖矿，该共识算法对于自私挖矿的抵抗力就越强。

（3）作恶收益：主要用于衡量双花攻击的成功率。该指标被量化为一定时间内区块链中双花收益的总金额，非法收益指的是攻击者无需消费就从商家那里获得的商品金额。我们假设每个诚实的区块都包含向商家的付款交易，当包含支付交易的区块达到σ确认后（比如比特币中$\sigma = 6$）或者攻击者选择放弃攻击该区块时，商家就可以提供服务或者商

品。在前一种情况，如果之后付款交易无效，则对于在确认后孤立的每个区块，攻击者将以出块奖励为单位接受双花奖励V_{ds}，也就是说如果攻击者成功双花攻击使得k个区块成为孤块，那么双花奖励将被定义为：

$$R_{ds}(k,\sigma,V_{ds})=\begin{cases}0, & k<\sigma\\ (k+1-\sigma)V_{ds}, & k\geq\sigma\end{cases},$$

其中$k+1-\sigma$是孤立的确认块的个数。此外，如果在到达σ确认之前，就使交易无效，则$R_{ds}=0$。攻击者并不会因为双花攻击失败而受到损失，因为商家还是会提供商品或服务，从而补偿了攻击者的损失。该指标能够从多个方面衡量共识协议是否能抵抗双花攻击，首先$\sum R_{\alpha}$会迫使攻击者去平衡失去出块奖励的风险和双花攻击成功得到的双花收益。第二，如果在σ确认之前广播冲突的交易，则允许商家延迟交货，以抵消攻击。第三，更长的分叉会在现实中带来更大的伤害，对攻击者来说回报也就更高。攻击者的作恶收益定义如下：

$$S(\alpha,\sigma,V_{ds})=\max_{s}\lim_{t\to\infty}\frac{\sum R_{a}+\sum R_{ds}}{t}-\alpha,$$

其中，t表示持续时间，是用出块间隔来衡量，α是没有双花攻击的平均出块奖励。在理想情况下，攻击者应该诚实遵守协议以避免丢掉出块奖励，但是如果V_{ds}足够大，攻击者就能被该激励诱惑。

（4）审查敏感性：主要用于衡量对审查攻击的抵抗性。该指标被量化为因为拒绝审查者的要求，导致诚实矿工的收益损失的最大百分比。我们选择不考虑攻击者的经济损失，因为如果审查威胁成功，则不会进行报复。只要其他诚实矿工能够坚定攻击者的决心，那么影响攻击者战略的唯一因素就是诚实矿工不进行合作的预期损失。与羽毛分叉攻击(Feather-forking attack,或惩罚性分叉来审查链上的交易。补充来说，这与生成分叉币种、空投或盈利无关，而与阻止网络参与者在区块链中执行交易操作有关。这样的攻击向量也要比 51%攻击更先进，因为它并不需要大多数的算力来将一个网络参与者列入黑名单当中)不同，在该评估标准中，攻击者在收到包含目标交易的区块后就会立刻开始报复，一旦合法矿工开始挖掘该区块，攻击者就会发起攻击。此设置很实用，因为攻击者可以通过一直监视网络中的交易立即进行挖矿。审查攻击和羽毛分叉的另一个区别是，允许攻击者放弃落后的链并能够在任何时间孤立下一个诚实的块。由于攻击者的目标是最大程度地增加诚实矿工的损失，因此攻击者并不会在落后的链上进行挖矿。因此在落后的链上进行挖掘并非总是最佳的。我们还考虑了多种攻击情形，例如在极端的攻击形式中，攻击者通过用空白区块替代诚实区块，从而延迟所有交易的确认时间，降低系统的可用性。审查敏感性定义如下：

$$C(\alpha)=\max_{s}\lim_{t\to\infty}\frac{\sum O_{c}}{\sum O_{c}+\sum R_{c}},$$

其中$\sum O_c$是诚实矿工因为审查攻击而遭受的累积奖励损失，以区块奖励为单位。在理想情况下，$C(\alpha)=0$，即诚实矿工没有拒绝审查请求的风险。

通过以上四个指标，可以对共识算法实现过程做安全审查，若无法同时满足以上指标，则该共识算法在实现过程较容易受到攻击。

3.2.2 渗透测试

“渗透测试”是完全模拟黑客可能使用的攻击技术和漏洞发现技术，对目标系统的安全做深入的探测，发现系统最脆弱的环节。渗透测试和黑客入侵最大区别在于渗透测试是经过客户授权，采用可控制、非破坏性质的方法和手段发现目标和网络设备中存在弱点，帮助管理者发现 IT 系统所面临的安全威胁。共识算法渗透测试的测试点主要有：

（1）公共参数

许多共识算法对如区块大小、出块间隔、奖励、主节点等都做了详细的规定，这些公共参数不应该被轻易修改，渗透测试应检查是否有违反公共参数的可能性。

（2）一致性/有效性/可用性/公平性

节点是否能在有限的时间内对共识的数据达成一致，所达成的数据是否合法，数据是否实时可用，节点是否能够公平的参与共识等。

（3）数据传输

共识通信多依赖 P2P 网络，需测试节点间网络是否能够正常连通、是否存在被 DDoS 攻击的可能、节点身份是否会被伪造、通信是否会被劫持等，如果通信有保密需求，则必须验证数据的加密和解密是否正常。

（4）API 测试

API 测试就是要检查应用程序与区块链生态系统之间的交互。这样做是为了验证 API 发送的请求和响应，并确保它们已正确格式化和执行。

（5）集成测试

由于跨不同环境和并行系统部署了区块链测试，因此对集成测试的需求增加。完成测试是为了确保不同组件之间能够无缝通信。测试团队对 API 进行测试，确保可以在验证阶段使用这些 API。

（6）性能测试

通过性能测试可确定潜在的瓶颈，并检查应用程序是否已准备好投入生产。确定性能的测试自动化是检查共识系统整体可扩展性的关键。

（7）安全测试

目的是确保完全保护区块链应用程序免受病毒和恶意程序等攻击。区块链的安全测试需要非常彻底同时应迅速响应。由于正在进行的事务无法停止，因此测试过程应足够有效以发现所有潜在威胁。有效的安全测试还有助于改善公司在消费者面临风险之前撤销有缺陷的商品的流程，有助于实现数字质量保证。

渗透测试通过有效利用编码错误，以潜在黑客的心态开展工作。用简单的话来说，测试人员本身就是黑客，并试图闯入网络以检测和报告安全漏洞。渗透测试人员花费的总时间取决于网络规模及其架构的复杂性。较小的测试只是时间问题，而较长的测试可能需要长达数周的时间。需要区块链渗透测试作为解决方案的一些挑战是：

- 缺乏测试工具
- 知识不足
- 不称职的策略
- 不可逆转的交易
- 性能和负载测试

有效的区块链测试可帮助组织通过连接的基础架构安全地构建和利用该技术。测试过程包括核心测试策略和服务，如云测试服务、功能测试、API 测试、集成测试、安全测试和性能测试。它还包括特定于区块链的测试策略，如块测试、智能合约测试和对等/节点测试。

3.2.3 渗透测试工具和代码安全审计

测试人员选择最合适的区块链渗透测试工具以减轻漏洞并提供最佳质量的结果同样重要，以下几个框架是常用的测试框架：

（1）Truffle 框架

Truffle 是最受欢迎的开发环境之一，也是用于区块链测试的测试框架。Truffle 为智能合约提供了简单的生命周期管理，包括对库链接、自定义部署和复杂的基于区块链的应用程序的支持。Truffle 还提供自动合同测试，开发人员可以使用 JavaScript 和 Solidity 编写自己的自动测试代码。它的一些显着特征是：

- 开发期间立即重建资产
- 可配置的构建管道，完全支持自定义构建过程
- 可编写脚本的部署和迁移框架
- 通过交互式控制台进行直接合同通信

（2）Embark

Embark 提供了一种简单的声明性方法定义要部署的智能合约及其相关性。以太坊测试仪为各种区块链测试需求提供可管理的 API 支持，旨在改善用户和开发人员的体验，并帮助他们轻松管理和执行所选工具。

（3）Populus

这里的测试由 python 测试框架提供支持，并提供了用于测试智能合约的有用实用程序。

代码审计是直接对程序的源代码进行检查，以确定程序源代码是否存在安全隐患，或者有编码不规范的地方，通过自动化工具或者人工审查的方式，对程序源代码逐条进行检查和分析，发现这些源代码缺陷引发的安全漏洞，并提供代码修订措施和建议。

代码审计对于共识系统的发展具有重要意义：一方面，代码审计可以节约安全投入，降低修复成本。研究表明，当应用发布后再执行代码修复，修复成本大约是设计编码阶段的 30 倍。所以，变被动防护为主动防御，从源头上控制安全隐患，可以最大程度节约成本；另一方面，代码审计可以降低系统安全风险。通过代码审计及时对代码层缺陷进行修复，从而大幅度提升系统整体安全性，避免巨额经济损失。

越来越多的共识系统安全事件正在倒逼代码审计的发展。目前，智能化代码审计，即利用计算机进行稳健性检验是代码审计最重要的方式，但掌握该项技术标准的国内公司并不多。代码审计亟待进一步普及与发展。

3.3 共识算法安全 CheckList

包含 2.1 和 2.2 的安全检查列表。结合 2.1 和 2.2 内容，共识算法安全检查包含以下方面：

（1）对算法参数进行检查，包含以下 3 个指标：

a.出块时间和难度调整窗口：检查出块间隔和难度调整窗口是否合适，若出块速率过快，那么受到网络延迟的影响，矿工通常会在这些块被传播之前就找到冗余块，较容易出现分叉。若出块速率过慢，那么就增加了用户等待事物确认的时间。

b. 块大小限制：区块大小和区块包含的交易数量紧密相关，通过检查区块大小防止交易量过大造成的合法用户阻塞问题。

c.货币政策：检查货币政策是否过于紧缩，若紧缩会导致矿工挖矿积极性不高，初始代币不流通。

（2）对算法进行模拟检查：通过 MDP 建模模拟挖矿过程，包括区块链状态、攻击者动作、随机转移矩阵和奖励矩阵。诚实矿工会根据自己的算力，以概率分布开采新的区块，而攻击者会决定是否要发布新区块、发布多少区块以及要对哪条链进行挖掘，而这些事件

都会触发 MDP 状态转换。并且对双花攻击和日蚀攻击建模，考虑这些攻击的最坏情况，模拟攻击者的攻击过程，可以及时发现共识协议的弱点。

（3）对算法实现过程进行检查：包含以下 4 个指标：

a. 链质量：主要观察算法能够替换诚实主链的难度，用区块链中诚实矿工挖出的区块数量除以诚实矿工和攻击者共同挖出的区块数量计算得出。若该值越大，则算法被攻击者替换诚实主链的难度就越大。

b. 激励兼容性：主要观察算法对自私挖矿的抵抗力，用网络中诚实矿工获得的奖励除以诚实矿工和攻击者总共获得的奖励计算得出。若该值越大，则诚实矿工获得的奖励更多，算法对于自私挖矿的抵抗力越大。

c. 作恶收益：主要观察算法对双花攻击的抵抗力，量化为一定时间内双花攻击获得的总收益。若该值越大，则算法对双花攻击的抵抗力更弱。

d. 审查敏感性：主要观察算法对审查攻击的抵抗力，量化为因为拒绝审查者的要求，导致诚实矿工的收益损失的最大百分比。若该值越大，则算法对审查攻击的抵抗力越弱。

针对共识机制的安全风险有很多，下表列举了主要风险和攻击类型，通过逐条对照可以对所设计的共识算法进行安全性评定。下表可作为共识算法安全的 Checklist。[2，3，8]

攻击类型	攻击对象	攻击手段	影响
51%攻击	PoW PoS DPoS	当攻击者能够控制区块链中超过 50%的能力（如挖掘能力或验证能力）时，他/她就能够进行恶意活动，例如双重支出或阻止其他节点接收其诚实交易。不同算法在攻击行为上有所不同，51%的攻击也是不可避免的。	
分叉攻击	PoW PoS	一个或多个节点通过控制全网特定百分比以上算力/数字资产，利用这些算力/数字资产隐秘计算新区块（攻击区块），构造区块链分叉，并在攻击区块达到一定长度之后向所有节点释放，迫使节点放弃原区块	篡改分叉后攻击者账户数据，实现双重支付，可导致链上记录回滚（可达数月）
女巫攻击	PoW	攻击者生成大量攻击节点并尽可能多的将攻击	实现攻击者对区

	PoS DPoS	节点植入网络中，在攻击期间，这些被称为女巫节点的攻击节点将只传播攻击者的块，导致攻击者算力无限接近于 1	块链网络的高度控制权
贿赂攻击	PoS	攻击者购买商品或者服务，商户开始等待区块链网络确认交易，此时攻击者开始在网络中首次宣称，对目前相对最长的不包含本次交易的主链进行奖励。当主链足够长时，攻击者开始放出更大的奖励，奖励那些在包含此次交易的链中挖矿的矿工。一旦确认达成后，放弃奖励。货物到手，同时放弃攻击者选中的链条。	以小于货物或者服务费用的成本获利
预计算攻击	PoS	将某一时间段内计算出的新区块扣留不公开，等到挖到第二块新区块后同时公布	攻击者所在分叉成为最长链
Sybil 攻击	PoW PoS DPoS	攻击者尝试通过在区块链中创建许多欺诈性身份来控制对等网络。这些身份似乎是唯一的用户或节点，实际上是由攻击者控制的。这些身份用于获得投票权，阻止验证权，甚至在区块链的社交网络中传播假消息。	使攻击者对网络有不成比例的控制权或包围诚实的节点，并试图影响到达该节点的信息，然后逐渐影响分类帐。
双花攻击	PoW PoS DPoS	当一个人尝试两次在区块链上花费特定金额的钱时，就会发生双重支出攻击[6]。当攻击者试图创建一个正常交易以包含在一个区块中，然后在一段时间后，创建一个欺诈性冲突交易并将其推到一个新的分叉欺诈性区块中	形成欺诈分支，影响交易

3.4.1 共识算法的安全性测试步骤：

针对共识算法的安全测试应遵循如下步骤：

a) 应保证来自于系统正常运行节点的请求能在规定时间内达成一致的、正确的共识最终

能够被系统接收并处理，需要测评是否能输出正确结果；

b) 应保证任意不超过理论值的节点数故障不会影响整个系统正常工作，需要测评诚实的节点是否会对相同的请求做出明确且一致的判断；

c) 应保证验证规则满足概然性与统一性，如共识机制中包含验证过程，需要测评每一参与节点是否具有独立判断能力；

d) 应保证共识方案的发布需伴随清晰的应用场景和规模参数，便于对机制的安全适配范围进行规范化的校验，主要测评该共识机制是否能够防止双重支付、女巫攻击等。

e) 应保证在理论值范围内的恶意节点对系统发出伪造、重复的恶意请求时，系统能够做出正确的响应，保证一致性。

4 共识算法安全案例分析

本节利用上一节的测试方法和标准，对共识算法安全的典型安全进行分析，并给出分析过程和结论。本节的案例包括现实中的案例和设计的案例两类，目标是涵盖前文介绍的全部安全威胁和测试方法。

下面列举部分现实的共识算法安全案例：

4.1 51% 攻击

2013 年 6 月，一种以莱特币为原型的创造的加密货币 Feathercoin 遭遇 51%攻击，攻击者通过双花攻击卷走了价值 6.38 万美元的代币，Feathercoin 的创始人表示此次攻击的算力可能来自于莱特币矿池。2014 年 7 月，比特币矿池 GHash.IO 在某天内获得了超过 51%的哈希率，引发了外界对比特币漏洞的担忧，尽管没有发生恶意攻击，但随后矿工逐渐离开矿池并且该矿池于 10 月关闭。2016 年 8 月，一群被称为“51crew”的攻击者劫持了两个基于以太坊的区块链，Krypton 和 Shift，通过发起双花攻击获取了价值 21465 Kryptons 的数字货币。Krypton 项目的代码复制于以太坊，拥有几乎一样的功能，这也被证实一些算力低的山寨币更容易受到 51%攻击。在 2018 年 4 月和 5 月，Verge 经历了两次 51%攻击，遭受了 180 万美元的价值损失。Verge 为了避免算力的集中化采用 5 种算法，项目在 5 种算法间切换，而攻击者正是利用 Verge 算法的漏洞，修改了区块链的时间戳，从而成功攻击。2018 年 5 月，一群比特币矿工获得了超过 50%的算力，对比特黄金发动了攻击，盗取了价值 1800 万美元的比特币。随后在 6 月，还攻击了其他四种加密货币，分别是 Monacoin，Zencash，Verge 和 Litecoin Cash。这些攻击者甚至不需要购买硬件矿机，只需要在攻击期间从算力租赁市场租赁显卡算力，这种方式极大地降低了攻击者攻击的成本。

同样的，在 2019 年 1 月，ETC 被发现受到了 51%攻击，攻击者总共获得了价值 110 万美元的以太币，攻击者同样不需要投资任何硬件设备。

从以上案例可见 51%攻击可分为以下几种情况：

（1） 与其它区块链公用相同的挖矿算法，导致攻击者可以从其它区块链抽调算力进行攻击，这多次发生在分叉币上；

（2） 矿池算力过于集中，矿池越大，矿工收益就越稳定，所以矿工倾向于选择大的矿池，导致矿池所拥有的算力越来越多，最终超过 51%，但该情况威胁一般不大，当矿池算力过大时，理性矿工会选择离开矿池，以使矿池算力恢复到一个合理的水平；

（3） 挖矿算法存在漏洞，导致矿工可以通过漏洞提高自己的算力；

（4） 攻击者可以通过算力租赁的方式使自己的算力超过 51%，这种情况一般对总算力较少的区块链有效；

（5） 专用硬件的出现导致拥有新硬件的攻击者可以以极低的成本拥有更高的算力；

（6） 出块间隔等参数设置不合理、发生自私挖矿攻击等，导致诚实矿工的算力大部分浪费在分叉上，攻击者可以以实际低于 51%的算力完成攻击；

从攻击者目的来看，一般有三种情况：（1）通过制造双花来使自己收益；（2）通过拒绝其它矿工的出块来增加挖矿收益；（3）竞争币单纯的对对手进行破坏。

4.2 DDoS 攻击

DDOS 的一种攻击形式是恶意节点通过发起大量尘埃交易以使得合法用户支付更高的交易费。2017 年 6 月，香港交易所 Bitfinex 遭受了 DDoS 攻击，导致其临时暂停。同年的 11 月 11 日，比特币交易池中被发现存在大小超过 11.5 万笔的未经确认的交易，导致价值 7 亿美元的交易停滞。2018 年 6 月，比特币的交易池再次遭到 4500 笔未经确认的垃圾交易攻击，使得交易池大小增加了 45MB，规模增加导致交易费飙升，合法用户被迫支付更高的费用完成交易。在 2019 年 11 月，比特币交易池达到了 2018 年 1 月以来的最高水平，交易池大小超过 90MB，垃圾交易量大幅上涨。此外，攻击者还会对网站发动 DDoS 攻击，使得系统卡顿、宕机，交易吞吐量减少。2018 年 7 月，一款运行在以太坊网络的游戏 Fomo3D 遭受黑客 DDoS 攻击，使得 24 小时内 Fomo3D 流量减少 38.32%，随后 Fomo3D 使用的安全管理网站开启高防验证，用户需等待 5 秒才能访问网站，这几乎是该网站的最高级 DDoS 防御策略。2020 年 2 月，加密货币交易所 Bitfinex 和 OKEx 均遭受多轮 DDOS 攻击，该攻击通过轰炸信息使得平台过载而停止服务，平台的服务被迅速关闭并进行维护。

从上述案例可见目前主要有以下几种攻击行为：

（1） 通过发送大量的垃圾交易使正常交易无法被及时处理，影响系统的活性；

（2） 通过对特定节点进行 DDoS 攻击，使节点无法正常同步及发送区块，造成网络分片，造成节点的不一致，影响共识的安全性；

（3） 针对运行于区块链网络上的特定应用进行 DDoS 攻击，导致应用不可用。

防范 DDoS 攻击可以从设置合理的交易费用、构建健壮的网络、使用更难预测的随机数进行出块节点选择等措施着手。

4.3 BGP 劫持攻击

在过去的几年中，托管采矿池或加密货币交易所的自治系统都遭受了 BGP 劫持攻击。2014 年，加拿大的一个恶意互联网服务提供商（ISP）公开了一些主要 ISP 的 BGP 前缀，这些 ISP 包括 Amazon、OVH、Digital Ocean、LeaseWeb 和 Alibaba，并且拦截了到矿池的流量，攻击者因此获得了 83,000 美元。2018 年 4 月，攻击者针对以太坊钱包 MyEtherWallet.com 发起了 BGP 攻击，这是一个用于在线交换以太坊令牌的开源 Web 应用程序。攻击者控制了以太坊钱包的域名系统，使得用户被重定向到了一个钓鱼网站，当用户将私钥输入到钓鱼网站时，攻击者将设法窃取其私钥。此次攻击持续了几个小时，最终攻击者窃取了 152,000 美元。

这类攻击的漏洞一般来自于底层网络服务提供商，选择可信的网络服务提供商是解决该类攻击的关键。

5 共识算法安全的最佳实践

对于实际部署的区块链系统，特别是基于开源区块链平台的区块链应用系统，共识算法通常是以一个可选模块的方式实现的，因此本节主要提供共识算法的具体实现的最佳实践，即区块链应用如何选择、评估、整合、使用共识算法实现，以及如何进行参数设置等等，从实际考虑，最佳实践围绕主流的区块链平台和典型的行业应用展开。

5.1 超级账本的共识算法安全最佳实践

超级账本（Hyperledger）是一个旨在推动区块链跨行业应用的开源项目，提供企业级的开源分布式账本框架和代码库。超级账本对传统区块链模型进行了革新，其中包括管理参与者的访问许可权，也就是超级账本是有权限的共享账本，为身份识别、审核及隐私提供了一个安全、健康的模型。

Hyperledger Fabric 是超级账本项目下正在孵化的一个项目，是分布式账本平台的实现。该平台利用熟悉和成熟的技术来运行智能合约，并且设计为模块化结构，允许组件如共识算法和成员服务模块插入即用。本节主要对 Hyperledger Fabric 的共识算法实践进行说明。Hyperledger Fabric 中使用了基于 Zookeeper 的 Apache Kafka 共识算法，该共识算法是 Fabric1.0 版本中提供的共识算法之一，之所以将 0.6 版本中的 PBFT 暂时取消，一是因为交易性能达不到要求，二是因为 Fabric 面向的联盟链环境中，所有节点都有准入控制，拜占庭容错的需求不是很强烈，反而是并发性能更重要。

Fabric 中的共识算法被分解为三个阶段：

（1）背书阶段：在可选择的节点上模拟交易并收集状态变化信息。客户端应用程序将构造一个交易提案来调用链码(chaincode)中的函数，该函数依次对账本状态进行读和/或写操作，然后根据背书策略将其发送给指定的背书节点。背书节点接收到交易提案后首先会验证提交者是否有权调用通道上的交易。之后背书节点执行链码，链码可以访问背书节点当前的账本状态，进行读取和写入，但是写入只是模拟写入，修改的是私有事务工作区，而不会记录到账本中。执行完成后背书节点调用 ESCC(Endorsement System Chaincode)对执行结果进行签名，然后响应给客户端应用程序。最后，客户端检查提案响应以验证其是否带有背书节点的签名。客户端会从不同的节点那里收集足够多的提案响应，以验证背书是否相同。由于每个节点可能在区块链的不同高度执行交易，因此提案响应可能会不同。在这种情况下，客户端必须将提案交给其他背书节点，以获得足够多相同的提案响应。

（2）排序阶段：接受提交的被认可的交易并进行排序，确保交易顺序的一致性。排序阶段通过排序服务(Ordering Service)提供的接口接受已经背书的交易，然后根据共识算法中的配置（如指定的配置信息中定义的时间限制或指定允许的交易数量），确保交易顺序和数量，然后将交易打包到区块中进行广播。排序服务可以以不同的方式实现，开发和测试阶段可以使用集中式排序服务。为了保证交易的机密性，交易服务不能看到交易中的具体内容，也就是交易内容可以使用哈希散列或加密方式去处理。

（3）验证阶段：获取有序块并验证其结果的正确性，包括检查背书策略和重复提交攻击。通道上的所有节点（包括背书节点和提交节点）都从网络中接收数据块。节点首先验证签名，之后每一个块中的所有交易都要通过 VSCC 验证，再通过 MVCC 验证。

VSCC（Validation System Chaincode）验证：验证系统链码（VSCC）会根据为链码指定的背书策略评估交易中的背书，如果不符合背书政策，则该交易被标记为无效。

MVCC（Multi-Version Concurrency Control）验证：该验证用于确保交易在背书阶段读取的多版本密钥要与提交阶段中本地的当前状态相同，类似于为了进行并发控制而进行的读写冲突检查，并且块中的所有有效交易要按顺序执行。如果读取的版本不匹配，表示之前的交易修改了读取的数据，则该交易被标记为无效。

潜在的验证失败主要分为两种：（1）语法错误：如无效输入、未验证的签名、重复交易等，此类交易要被丢弃；（2）逻辑错误：此类错误较为复杂，应该定义策略决定是继续执行还是终止。

Fabric 是用 Go 语言编写的，并使用 gRPC 框架在客户端、节点之间通信。Fabric 一般会配置如下模块：

（1）背书政策：背书政策主要包括在交易请求提交给订购者前，需要执行多少次交易并且签名，以便交易通过 VSCC 验证。交易背书的 VSCC 验证会根据收集到的背书评估背书政策，并检查可满足性，背书政策的复杂性将影响资源以及收集和评估它的时间。

（2）通道：通道将交易彼此隔离，提交给不同渠道的交易彼此独立地进行排序、交付和处理。通道为系统中交易处理的各个方面带来了固有的并行性。Fabric 中的每个通道都形成一个逻辑区块链，通道的配置在特殊块中的元数据中进行维护，每个配置块都包含完整的通道配置。

（3）Gossip 协议：由于大多数共识算法（如 CFT 和 BFT 共识机制）都受带宽限制，因此排序阶段的吞吐量受其节点的网络容量的限制，共识协议将无法通过添加更多节点达成更广泛的共识，反而共识范围会缩小，吞吐量会下降。但是在 Kafka 协议中，排序阶段和验证阶段是分离的，因此可以在排序阶段之后将执行结果更为有效地广播给所有节点以进行验证，这是 Gossip 协议解决的目标之一。除以之外，对新加入网络的节点以及长时间断开网络连接的节点进行状态转移，这也是 Gossip 协议的目标。Gossip 协议实现基于 gRPC，并且使用具有相互认证的 TLS，这使得节点可以将 TLS 凭证绑定到远程节点上。Gossip 协议维护网络中所有节点的实时信息，节点可以在网络中断后重新连接到该信息。

（4）排序服务(ordering service)：排序服务用于管理多个通道，并提供如下服务：原子广播，用于建立交易顺序；当配置更新交易被广播时修改通道配置；访问控制，对某些交易广播和块接受进行限制。排序服务由链上的创世块完成。

5.2 以太坊共识算法安全最佳实践

比特币是最早的 PoW 共识算法的实践，但随着比特币的发展，市场是逐渐出现了专业矿机专门针对哈希算法、散热、能耗进行优化，这脱离了比特币网络节点运行在普通计

算机中公平参与挖矿的初衷，并且会造成节点中心化，还可能带来51%攻击。因此，以太坊中改进了PoW算法。以太坊早期起草的共识算法中使用的是Dagger-Hashimoto算法，但是Dagger-Hashimoto算法容易受到共享内存硬件加速的影响，所以最后对Dagger-Hashimoto算法进行改进得到了Ethash算法，是Ethereum1.0中使用的共识算法，在Ethereum2.0中计划使用PoS共识算法。Ethash算法增加了对内存的需求，即在进行挖矿时，需要占用大量的内存空间，这是ASIC矿机不具备的，因此将挖矿算法从CPU密集型转为了IO密集型。

传统的PoW算法仅仅是不断尝试nonce，将该nonce与难度阈值进行比较，而以太坊的Ethash算法中求解nonce仅仅是第一步，还需要利用内存进行一些混合运算，然后将该值与阈值进行比较。算法首先需要一些数据集，数据集生成方式如下：

（1） 存在一个种子（Seed），该种子与区块高度有关；

（2） 根据种子计算出16MB的伪随机缓存cache，由种子计算出第一个数，之后的每个数都由前一个数取哈希得到，轻节点存储该cache。

（3） 根据cache生成1GB的dataset（DAG），dataset中每一项数据都由cache中的数据参与生成，循环迭代256次，全节点会存储cache和dataset。

矿工在挖矿时，首先尝试随机数nonce，在DAG中，利用区块头部和nonce计算出一个DAG中的初始位置A，然后读取A和A相邻位置的元素A’，再通过A和A’计算出另外两个位置B和B’，以此类推总共迭代64次，总共取出128个数，将其计算得出的哈希与目标阈值比较，若满足条件则挖矿成功，否则重新尝试nonce。

Cache和dataset中的数据并非不变，大概每隔30000个区块就会重新计算一遍。而cache的大小16MB和dataset的大小1GB每年也会增大一点。这里选择16MB的cache大小是因为若选取较小的缓存，则太容易继续使用ASIC矿机进行挖矿，而16MB需要比较高的缓存读取带宽。而更大的缓存会使得算法较难而轻节点客户端无法进行区块校验。选择1GB的dataset是为了要求内存级别超过大多数专用内存和缓存的大小，但是普通计算机还能使用。每隔30000块重新计算一遍是因为间隔太大会使得内存不经常更新而仅仅是经常读取，而间隔太小又会使算力较小的机器要花大量时间在更新数据集的固定成本上。

以太坊的共识算法是基于改进后的GHOST协议（Greedy Heaviest Observe Subtree），GHOST协议是为了解决以太坊中出块时间过快所导致的安全性问题。以太坊中出块时间为15秒，而比特币是10分钟，出块时间过快会导致暂时性分叉过多，如果只用普通的PoW算法，那么假如有多个矿工同时挖出区块，算力大的矿池通常地理位置较优越，其网

络与更多的节点相连，它发布的区块能更快地在网络中传播，成为主链的可能性也越大。这对算力小的节点很不公平，并且不利于算力小的节点达成共识，因此以太坊中引入了GHOST 协议。GHOST 协议就是在区块链中 7 代及其以内的叔父区块均可以得到奖励，而矿工若包含一个叔父区块也可以得到除了出块奖励额外的奖励。因此引入 GHOST 协议可以鼓励矿工挖矿，不会因为算力小而打消矿工积极性，并且解决了分叉过多难以达成共识的情况。

以太坊中的每个节点都在以太坊虚拟机（EVM）上运行，EVM 可以执行用以太坊编程语言 Solidity 编写好的复杂代码，Solidity 是一种类 JavaScript 的语言。使用 EVM 跨以太网网络进行并行计算会很慢并且成本较高，因此让每个节点都自己运行 EVM 可以在整个区块链中保持共识。

以太坊共有 4 个阶段，即前沿、家园、大都会和宁静，前三个阶段采用的都是 PoW 共识机制，而在第四阶段采用的是 PoS 共识机制，名为 Casper。PoS 旨在解决 PoW 的缺点：能源消耗过大；鼓励矿工投资专门的硬件，违背了去中心化的理念；存在自私挖矿等攻击。PoS 可以被描述为“虚拟挖矿”，矿工购买的是 ETH 而不再是硬件和电力，共识机制根据矿工持有的代币数量成比例地分配投票权，而不是算力。因为不再用算力进行挖矿，因此可以说挖矿几乎“不用付出任何代价”，那么矿工很可能会朝着多个方向同时挖矿以增加其收益，这样会不利于达成共识。因此 Casper 是要求矿工必须锁定他们的一些币作为保证金。当他们发现一个可以被加到链上的区块时，就会下赌注验证。如果区块成功上链，验证者可以得到相应比例的奖励。但是当验证者试图做一些恶意行为时，他们也会受到相应的惩罚。这样可以解决账本分叉的问题。

参考文献

[1] Fischer M J, Lynch N A, Paterson M S. Impossibility of distributed consensus with one faulty process[J]. Journal of the ACM (JACM), 1985, 32(2): 374-382.

[2] Gilbert S, Lynch N. Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services[J]. Acm Sigact News, 2002, 33(2): 51-59.

[3] Ben-Or M. Another advantage of free choice (Extended Abstract) Completely asynchronous agreement protocols[C]//Proceedings of the second annual ACM symposium on Principles of distributed computing. 1983: 27-30.

[4] Dwork C, Lynch N, Stockmeyer L. Consensus in the presence of partial synchrony[J]. Journal of the ACM (JACM), 1988, 35(2): 288-323.

[5] Leslie Lamport, Robert E. Shostak, and Marshall C. Pease. The Byzantine Generals Problem. ACM Trans. Program. Lang. Syst., 4(3):382–401, 1982.

[6] Lamport L. The part-time parliament[M]//Concurrency: the Works of Leslie Lamport. 2019: 277-317.

[7] Ongaro D, Ousterhout J. The raft consensus algorithm[J]. 2015.

[8] Castro M, Liskov B. Practical Byzantine fault tolerance[C]//OSDI. 1999, 99(1999): 173-186.

[9] Van Renesse R, Schneider F B. Chain Replication for Supporting High Throughput and Availability[C]//OSDI. 2004, 4(91–104).

[10] Guerraoui R, Knezevic N, Quema V, et al. Stretching bft[R]. 2010.

[11] Özsu M T, Valduriez P. Principles of distributed database systems[M]. Englewood Cliffs: Prentice Hall, 1999.

[12] Abd-El-Malek M, Ganger G R, Goodson G R, et al. Fault-scalable Byzantine fault-tolerant services[J]. ACM SIGOPS Operating Systems Review, 2005, 39(5): 59-74.

[13] Cowling J, Myers D, Liskov B, et al. HQ replication: A hybrid quorum protocol for Byzantine fault tolerance[C]//Proceedings of the 7th symposium on Operating systems design and implementation. 2006: 177-190.

[14] Guerraoui R, Knežević N, Quéma V, et al. The next 700 BFT protocols[C]//Proceedings of the 5th European conference on Computer systems. 2010: 363-376.

[15] Kotla R, Alvisi L, Dahlin M, et al. Zyzzyva: speculative byzantine fault tolerance[C]//Proceedings of twenty-first ACM SIGOPS symposium on Operating systems principles. 2007: 45-58.

[16] Abraham I, Gueta G, Malkhi D, et al. Revisiting fast practical byzantine fault tolerance[J]. arXiv preprint arXiv:1712.01367, 2017.

[17] Gueta G G, Abraham I, Grossman S, et al. SBFT: a scalable and decentralized trust infrastructure[C]//2019 49th Annual IEEE/IFIP international conference on dependable systems and networks (DSN). IEEE, 2019: 568-580.

[18] Singh A, Fonseca P, Kuznetsov P, et al. Zeno: Eventually Consistent Byzantine-Fault Tolerance[C]//NSDI. 2009, 9: 169-184.

[19] through Byzantine locking[C]//2010 IEEE/IFIP International Conference on Dependable Systems & Networks (DSN). IEEE, 2010: 363-372.

[20] A. Clement, E. Wong, L. Alvisi, M. Dahlin, and M. Marchetti, “Making Byzantine Fault Tolerant Systems Tolerate Byzantine Faults,” Symp. A Q. J. Mod. Foreign Lit., 2009.

[21] Y. Amir, B. Coan, J. Kirsch, and J. Lane, “Prime: Byzantine replication under attack,” IEEE Trans. Dependable Secur. Comput., 2011.

[22] G. S. Veronese, M. Correia, A. N. Bessani, and L. C. Lung, “Spin one’s wheels? Byzantine fault tolerance with a spinning primary,” in Proceedings of the IEEE Symposium on Reliable Distributed Systems, 2009.

[23] P. L. Aublin, S. Ben Mokhtar, and V. Quema, “RBFT: Redundant byzantine fault tolerance,” in Proceedings — International Conference on Distributed Computing Systems, 2013.

[24] Pass R, Shi E. The sleepy model of consensus[C]//International Conference on the Theory and Application of Cryptology and Information Security. Springer, Cham, 2017: 380-409.

[25] Baird L. The swirlds hashgraph consensus algorithm: Fair, fast, byzantine fault tolerance[J]. Swirlds, Inc. Technical Report SWIRLDS-TR-2016, 2016, 1.

[26] Satoshi, Nakamoto. "Bitcoin: A peer-to-peer electronic cash system." *Consulted* 1.2012 (2008): 28.

[27] Dwork C, Naor M. Pricing via processing or combatting junk mail[C]//Annual International Cryptology Conference. Springer, Berlin, Heidelberg, 1992: 139-147.

[28] Jakobsson M, Juels A. Proofs of work and bread pudding protocols[M]//Secure information networks. Springer, Boston, MA, 1999: 258-272.

[29] Back A. Hashcash-a denial of service counter-measure[J]. 2002.

[30] Sompolinsky Y, Zohar A. Secure high-rate transaction processing in bitcoin[C]//International Conference on Financial Cryptography and Data Security. Springer, Berlin, Heidelberg, 2015: 507-527.

[31] Eyal I, Gencer A E, Sirer E G, et al. Bitcoin-ng: A scalable blockchain protocol[C]//13th {USENIX} symposium on networked systems design and implementation ({NSDI} 16). 2016: 45-59.

[32] Pass R, Shi E. Fruitchains: A fair blockchain[C]//Proceedings of the ACM Symposium on Principles of Distributed Computing. 2017: 315-324.

[33] Sompolinsky Y, Lewenberg Y, Zohar A. SPECTRE: A Fast and Scalable Cryptocurrency Protocol[J]. IACR Cryptol. ePrint Arch., 2016, 2016: 1159.

[34] Li C, Li P, Zhou D, et al. Scaling nakamoto consensus to thousands of transactions per second[J]. arXiv preprint arXiv:1805.03870, 2018.

[35] Zhang R, Preneel B. Lay down the common metrics: Evaluating proof-of-work consensus protocols' security[C]//2019 IEEE Symposium on Security and Privacy (SP). IEEE, 2019: 175-192.
[36] Proof of stake [Online], available: https://en.bitcoin.it/wiki/Proof of Stake, April 11, 2018.
[37] King S, Nadal S. Ppcoin: Peer-to-peer crypto-currency with proof-of-stake[J]. self-published paper, August, 2012, 19: 1.
[38] Buterin V, Griffith V. Casper the friendly finality gadget[J]. arXiv preprint arXiv:1710.09437, 2017.
[39] Bentov I, Pass R, Shi E. Snow White: Provably Secure Proofs of Stake[J]. IACR Cryptol. ePrint Arch., 2016, 2016: 919.
[40] Kiayias A, Russell A, David B, et al. Ouroboros: A provably secure proof-of-stake blockchain protocol[C]//Annual International Cryptology Conference. Springer, Cham, 2017: 357-388.
[41] David B, Gaži P, Kiayias A, et al. Ouroboros praos: An adaptively-secure, semi-synchronous proof-of-stake blockchain[C]//Annual International Conference on the Theory and Applications of Cryptographic Techniques. Springer, Cham, 2018: 66-98.
[42] Badertscher C, Gaži P, Kiayias A, et al. Ouroboros genesis: Composable proof-of-stake blockchains with dynamic availability[C]//Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security. 2018: 913-930.
[43] Kerber T, Kiayias A, Kohlweiss M, et al. Ouroboros crypsinous: Privacy-preserving proof-of-stake[C]//2019 IEEE Symposium on Security and Privacy (SP). IEEE, 2019: 157-174.
[44] Garay J, Kiayias A, Leonardos N. The bitcoin backbone protocol: Analysis and applications[C]//Annual International Conference on the Theory and Applications of Cryptographic Techniques. Springer, Berlin, Heidelberg, 2015: 281-310.
[45] Miller A, Juels A, Shi E, et al. Permacoin: Repurposing bitcoin work for data preservation[C]//2014 IEEE Symposium on Security and Privacy. IEEE, 2014: 475-490.
[46] Park S, Pietrzak K, Kwon A, et al. Spacemint: A Cryptocurrency Based on Proofs of Space[J]. IACR Cryptol. ePrint Arch., 2015, 2015: 528.
[47] Cohen B, Pietrzak K. The chia network blockchain[J]. 2019.
[48] Benet J, Greco N. Filecoin: A decentralized storage network[J]. Protoc. Labs, 2018: 1-36.
[49] Zhang F, Eyal I, Escriva R, et al. {REM}: Resource-efficient mining for blockchains[C]//26th {USENIX} Security Symposium ({USENIX} Security 17). 2017: 1427-1444.
[50] Decker C, Seidel J, Wattenhofer R. Bitcoin meets strong consistency[C]//Proceedings of the 17th International Conference on Distributed Computing and Networking. 2016: 1-10.
[51] Kogias E K, Jovanovic P, Gailly N, et al. Enhancing bitcoin security and performance with strong consistency via collective signing[C]//25th {usenix} security symposium ({usenix} security 16). 2016: 279-296.
[52] Abraham I, Malkhi D, Nayak K, et al. Solida: A blockchain protocol based on reconfigurable byzantine consensus[J]. arXiv preprint arXiv:1612.02916, 2016.

[53] Pass R, Shi E. Hybrid consensus: Efficient consensus in the permissionless model[C]//31st International Symposium on Distributed Computing (DISC 2017). Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2017.

[54] Pass R, Shi E. Thunderella: Blockchains with optimistic instant confirmation[C]//Annual International Conference on the Theory and Applications of Cryptographic Techniques. Springer, Cham, 2018: 3-33.

[55] Gilad Y, Hemo R, Micali S, et al. Algorand: Scaling byzantine agreements for cryptocurrencies[C]//Proceedings of the 26th Symposium on Operating Systems Principles. 2017: 51-68.

[56] Luu L, Narayanan V, Zheng C, et al. A secure sharding protocol for open blockchains[C]//Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. 2016: 17-30.

[57] Kokoris-Kogias E, Jovanovic P, Gasser L, et al. Omniledger: A secure, scale-out, decentralized ledger via sharding[C]//2018 IEEE Symposium on Security and Privacy (SP). IEEE, 2018: 583-598.

[58] Schindler P, Judmayer A, Stifter N, et al. Hydrand: Practical continuous distributed randomness[J]. 2020.

[59] Al-Bassam M, Sonnino A, Bano S, et al. Chainspace: A sharded smart contracts platform[J]. arXiv preprint arXiv:1708.03778, 2017.

[60] Zamani M, Movahedi M, Raykova M. Rapidchain: Scaling blockchain via full sharding[C]//Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security. 2018: 931-948.

[61] Miller A, Xia Y, Croman K, et al. The honey badger of BFT protocols[C]//Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. 2016: 31-42.

[62] Yin M, Malkhi D, Reiter M K, et al. Hotstuff: Bft consensus with linearity and responsiveness[C]//Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing. 2019: 347-356.

[63] Mazieres D. The stellar consensus protocol: A federated model for internet-level consensus[J]. Stellar Development Foundation, 2015, 32.

[64] 刘懿中, 刘建伟, 张宗洋, 等. 区块链共识机制研究综述[J]. 密码学报, 2019, 6(4): 395-432.

[65] Garay J, Kiayias A. Sok: A consensus taxonomy in the blockchain era[C]//Cryptographers' Track at the RSA Conference. Springer, Cham, 2020: 284-318.

[66] Pritchett D. Base: An acid alternative[J]. Queue, 2008, 6(3): 48-55.

[67] Goldreich O, Micali S, Wigderson A. How to play any mental game, or a completeness theorem for protocols with honest majority[M]//Providing Sound Foundations for Cryptography: On the Work of Shafi Goldwasser and Silvio Micali. 2019: 307-328.

[68] Canetti R. Universally composable security: A new paradigm for cryptographic protocols[C]//Proceedings 42nd IEEE Symposium on Foundations of Computer Science. IEEE, 2001: 136-145.

[69] 苏婷. UC 安全理论及应用研究[D]. 山东大学, 2009.

[70] Garay J, Kiayias A, Leonardos N. The bitcoin backbone protocol with chains of variable difficulty[C]//Annual International Cryptology Conference. Springer, Cham, 2017: 291-323.

[71] Kiayias A, Panagiotakos G. Speed-Security Tradeoffs in Blockchain Protocols[J]. IACR Cryptol. ePrint Arch., 2015, 2015: 1019.

[72] Pass R, Seeman L, Shelat A. Analysis of the blockchain protocol in asynchronous networks[C]//Annual International Conference on the Theory and Applications of Cryptographic Techniques. Springer, Cham, 2017: 643-673.

[73] Eyal I, Sirer E G. Majority is not enough: Bitcoin mining is vulnerable[C]//International conference on financial cryptography and data security. Springer, Berlin, Heidelberg, 2014: 436-454.

[74] Nayak K, Kumar S, Miller A, et al. Stubborn mining: Generalizing selfish mining and combining with an eclipse attack[C]//2016 IEEE European Symposium on Security and Privacy (EuroS&P). IEEE, 2016: 305-320.

[75] Carlsten M. The impact of transaction fees on bitcoin mining strategies[D]. Princeton University, 2016.

[76] Göbel J, Keeler H P, Krzesinski A E, et al. Bitcoin blockchain dynamics: The selfish-mine strategy in the presence of propagation delay[J]. Performance Evaluation, 2016, 104: 23-41.

[77] Houy N. The Bitcoin mining game[J]. Available at SSRN 2407834, 2014.

[78] Solat S, Potop-Butucaru M. Zeroblock: Preventing selfish mining in bitcoin[J]. arXiv preprint arXiv:1605.02435, 2016.

[79] Bahack L. Theoretical bitcoin attacks with less than half of the computational power (draft)[J]. arXiv preprint arXiv:1312.7013, 2013.

[80] Bonneau J, Miller A, Clark J, et al. Sok: Research perspectives and challenges for bitcoin and cryptocurrencies[C]//2015 IEEE symposium on security and privacy. IEEE, 2015: 104-121.

[81] Kroll J A, Davey I C, Felten E W. The economics of Bitcoin mining, or Bitcoin in the presence of adversaries[C]//Proceedings of WEIS. 2013, 2013: 11.

[82] Miller A, LaViola Jr J J. Anonymous byzantine consensus from moderately-hard puzzles: A model for bitcoin[J]. University of Central Florida Tech. Report CS-TR-14-01 (accessed 5 June 2019) https://socrates1024. s3. amazonaws. com/consensus. pdf, 2014.

[83] Gervais A, Karame G O, Wüst K, et al. On the security and performance of proof of work blockchains[C]//Proceedings of the 2016 ACM SIGSAC conference on computer and communications security. 2016: 3-16.

[84] Castro M, Liskov B. Practical Byzantine fault tolerance and proactive recovery[J]. ACM Transactions on Computer Systems (TOCS), 2002, 20(4): 398-461.

[85] K. Zheng, Y. Liu, C. Dai, Y. Duan and X. Huang, “Model Checking PBFT Consensus Mechanism in Healthcare Blockchain Network,” in the 9th International Conference on Information Technology in Medicine and Education (ITME), Hangzhou, 2018, pp. 877-881.

[86] H. Sukhwani, J. M. Martínez, X. Chang, K. S. Trivedi and A. Rindos, “Performance Modeling of PBFT Consensus Process for Permissioned Blockchain Network (Hyperledger Fabric),” in 2017 IEEE 36th Symposium on Reliable Distributed Systems (SRDS), Hong Kong, 2017, pp. 253-255.

[87] R. Halalai, T. A. Henzinger and V. Singh, “Quantitative Evaluation of BFT Protocols,” in the 8th International Conference on Quantitative Evaluation of SysTems, Aachen, 2011, pp. 255-264.